

Table of Contents

Introduction to Android Development



This is where you can find all the labs for **Android 101**.

We will be covering and introduction to Android using a simple Case Study called ***Donation***

The labs will begin with a simple Android app and progress on to a fully fledged App with Custom Adapters and built-in persistence.

If you're looking for a bit more than an introduction, you should have a look at my Mobile Application Development Labs.

Lab 01: 'Hello World'

This lab is really just to get you up to speed with using the Android environment and the basic structure of an Android Application. We'll put our own spin on the classic 'Hello World' application and demonstrate the use of some very basic widgets and event handling.

Objectives

- Be able to download and install Android Studio, the Android SDK and understand its key features. In particular:
  - Project View
  - SDK Manager
  - AVD Manager
- Be able to understand the structure of an Android Studio Project
- Have created a simple Android App (HelloWorld), and be able to manage it within the Android Studio environment.

Setup

Instructions for working in Walton Building PC Labs:

If you are working on the workstations in the IT Building, Android Studio should already be installed so proceed with the next step of the lab.

Instructions for working on your own laptop

Download and install Android Studio from

- https://developer.android.com/sdk/index.html

Select the correct version for your OS. The primary prerequisite for installing Android Studio is that you have a recent Java installation on your workstation. To see if you have Java, and to install it if you do not, visit:

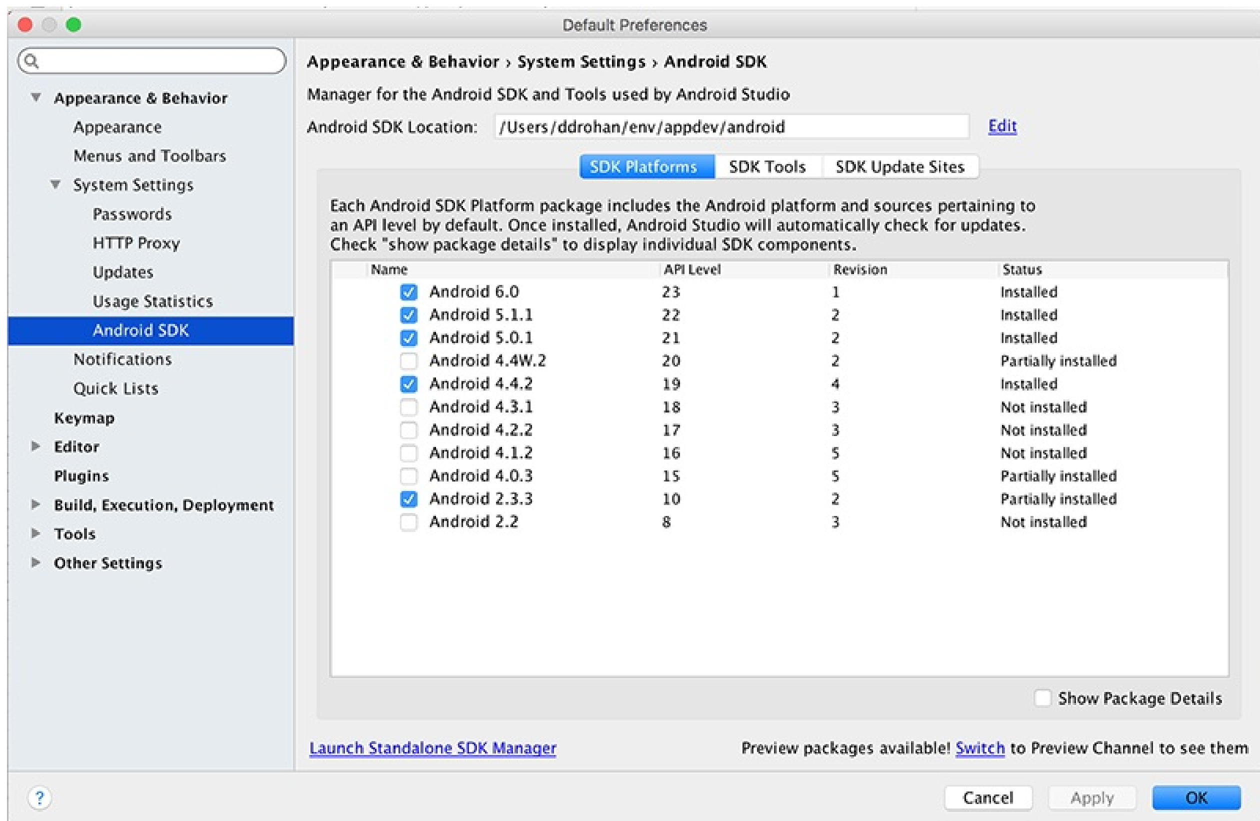
- http://www.java.com

Android Studio looks after downloading the Android SDK for you but you can download and install the Android SDK separately from

- https://developer.android.com/sdk/installing/index.html

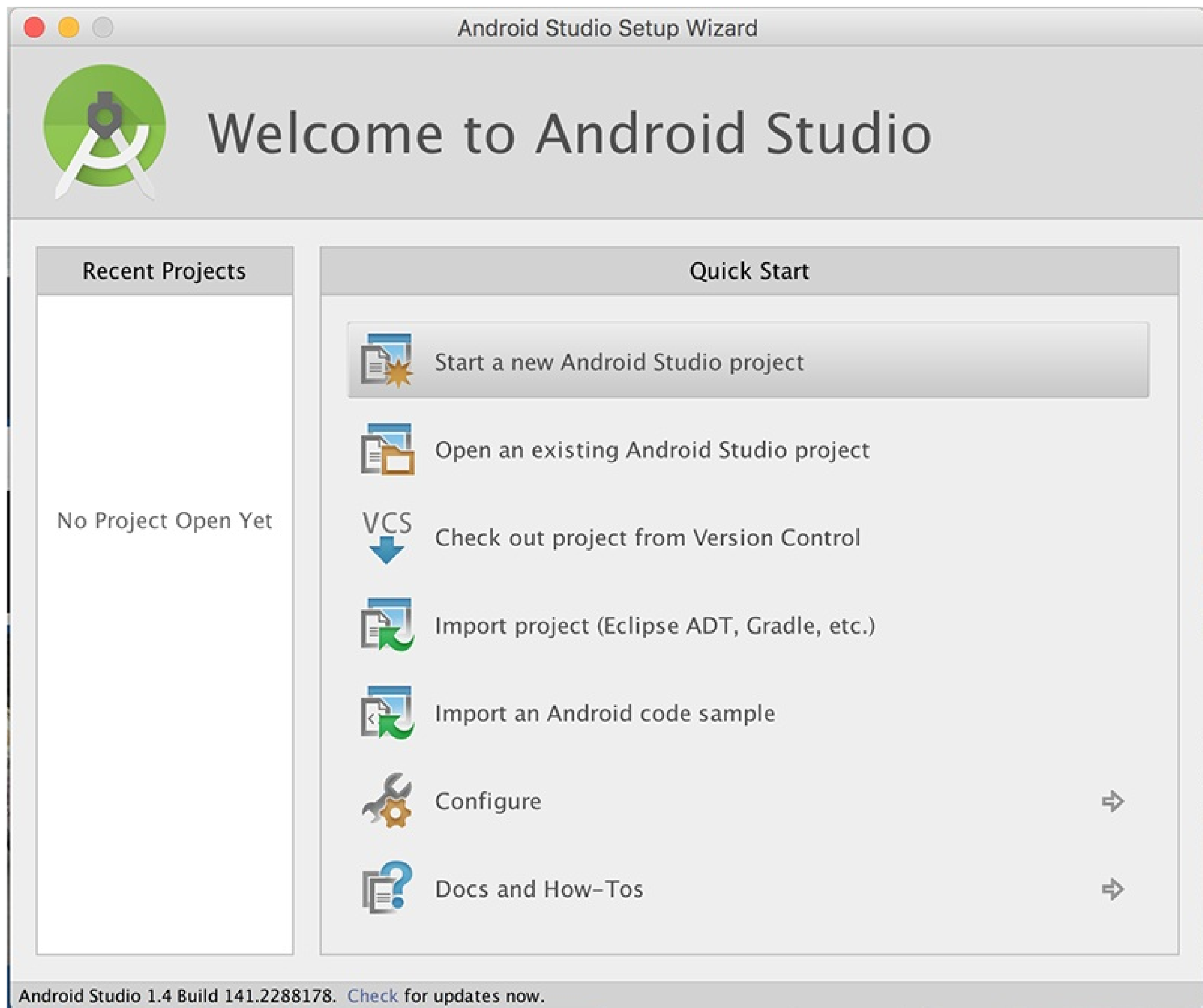
This download is actually the SDK Manager (not the full SDK), where you can choose which versions of the Android platform you want to install & develop with - select **anything above API 21 (Version 5.0)**. (But I'd leave all the TV and Wearable stuff :))

NOTE : these downloads are quite large so it's advisable to have these versions installed **BEFORE** class.


Default Preferences
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
Passwords
HTTP Proxy
Updates
Usage Statistics
Android SDK
Notifications
Quick Lists
Keymap
Editor
Plugins
Build, Execution, Deployment
Tools
Other Settings
Appearance & Behavior › System Settings › Android SDK
Manager for the Android SDK and Tools used by Android Studio
Android SDK Location: /Users/ddrohan/env/appdev/android
Edit
SDK Platforms
SDK Tools
SDK Update Sites
Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.
Name
API Level
Revision
Status
Android 6.0 23 1 Installed
Android 5.1.1 22 2 Installed
Android 5.0.1 21 2 Installed
Android 4.4W.2 20 2 Partially installed
Android 4.4.2 19 4 Installed
Android 4.3.1 18 3 Not installed
Android 4.2.2 17 3 Not installed
Android 4.1.2 16 5 Not installed
Android 4.0.3 15 5 Partially installed
Android 2.3.3 10 2 Partially installed
Android 2.2 8 3 Not installed
Show Package Details
Launch Standalone SDK Manager
Preview packages available! Switch to Preview Channel to see them
Cancel
Apply
OK

First Android Project - "HelloWorld"

In Android Studio, select File->New->New Project, or if it's a first run, select "Start a new Android Studio Project"



Press "Next" (or click the option) and then give the project a name: 'HelloWorld'

It's recommended you change the default package name also and it's probably worth changing the Project Location too but you can take the default for the moment.

Select the Platform(s) you want your app to run on - we'll just stick with Phone & Tablet and choose an appropriate Minimum SDK.

Create New Project

Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

- [x] Phone and Tablet

Minimum SDK: API 15: Android 4.0.3 (IceCreamSandwich)

Lower API levels target more devices, but have fewer features available. By targeting API 15 and later, your app will run on approximately 94.0% of the devices that are active on the Google Play Store.

Help me choose

- [ ] Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

- [ ] TV

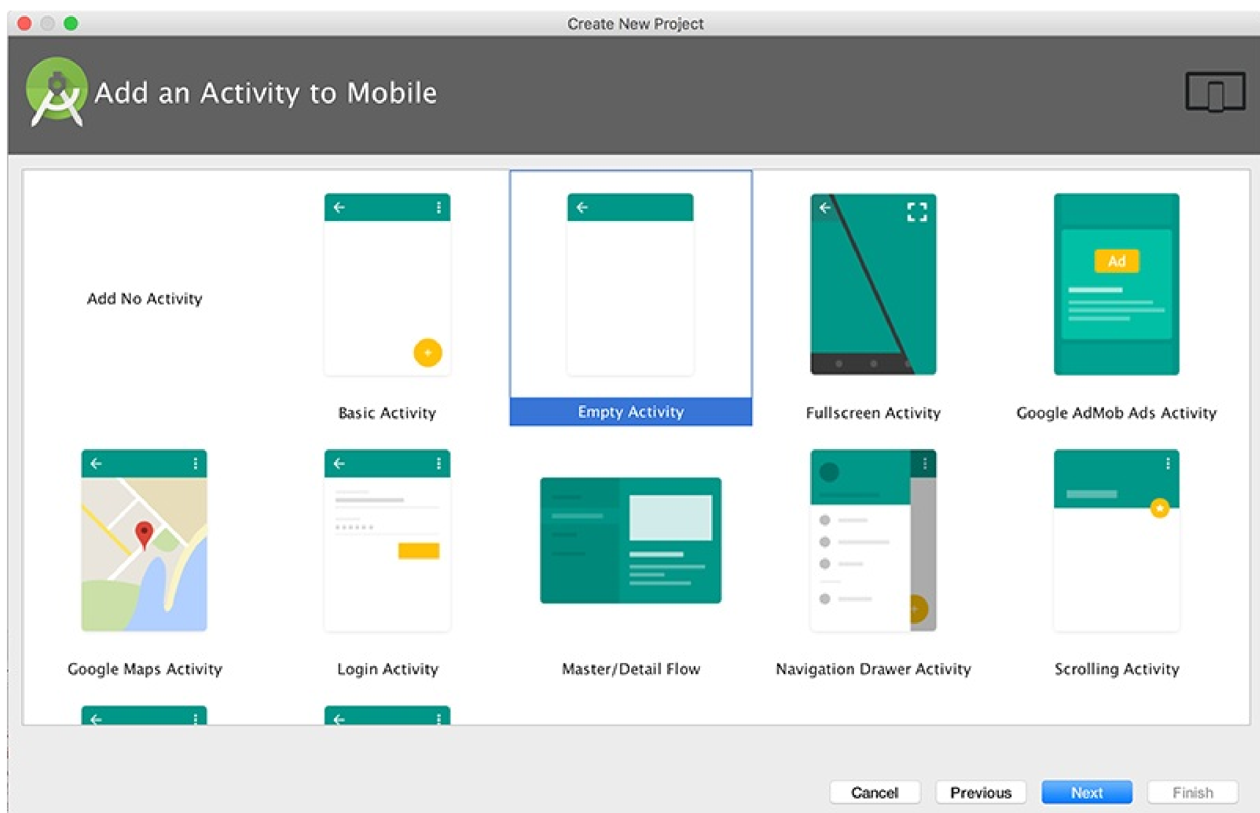
Minimum SDK: API 21: Android 5.0 (Lollipop)

- [ ] Android Auto
- [ ] Glass (Not Installed) Download

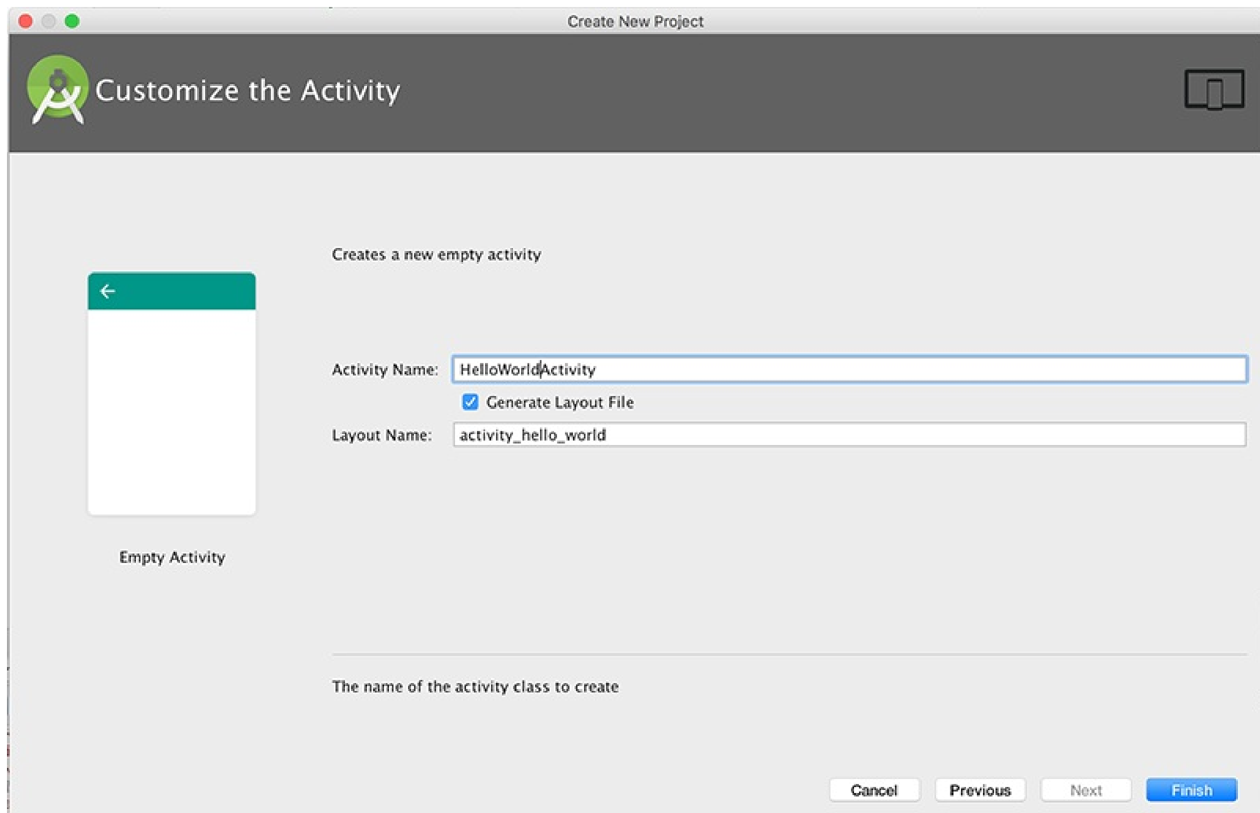
Minimum SDK

Cancel | Previous | Next | Finish

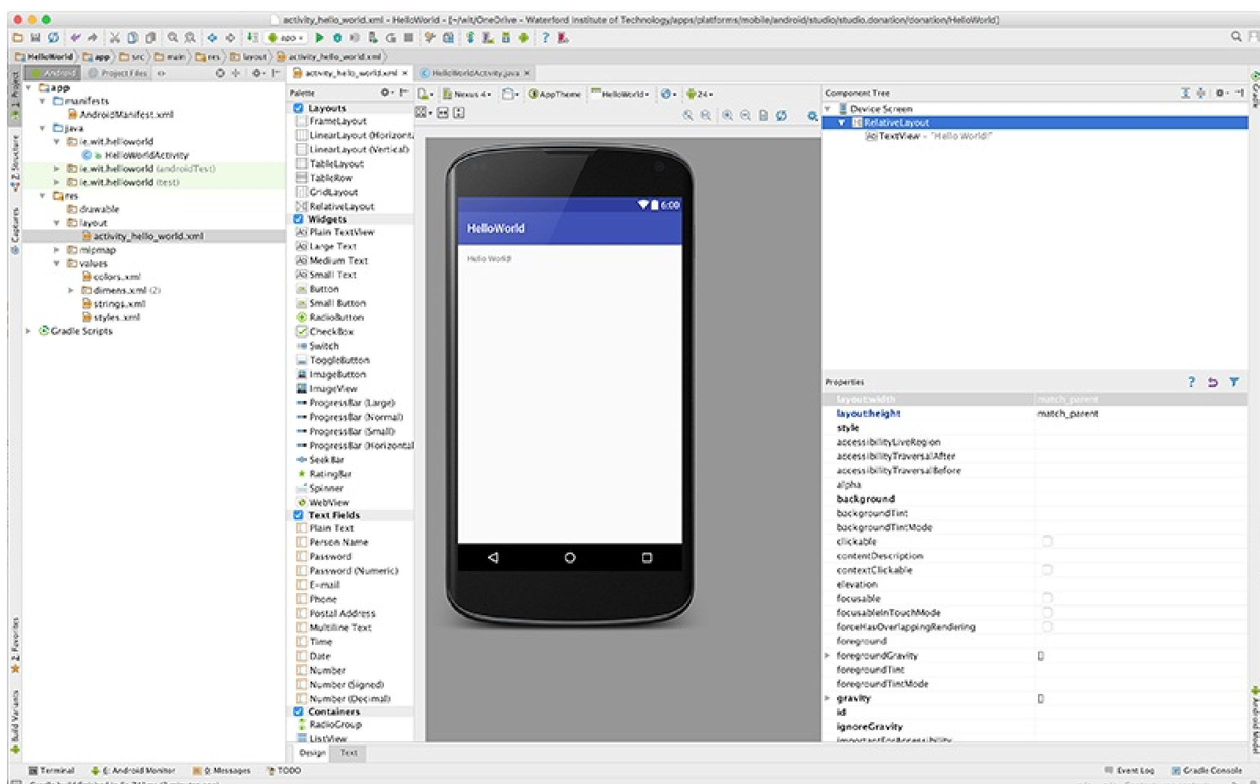
You should choose an Empty Activity as your activity type on the next screen



and name it as in the screenshot below

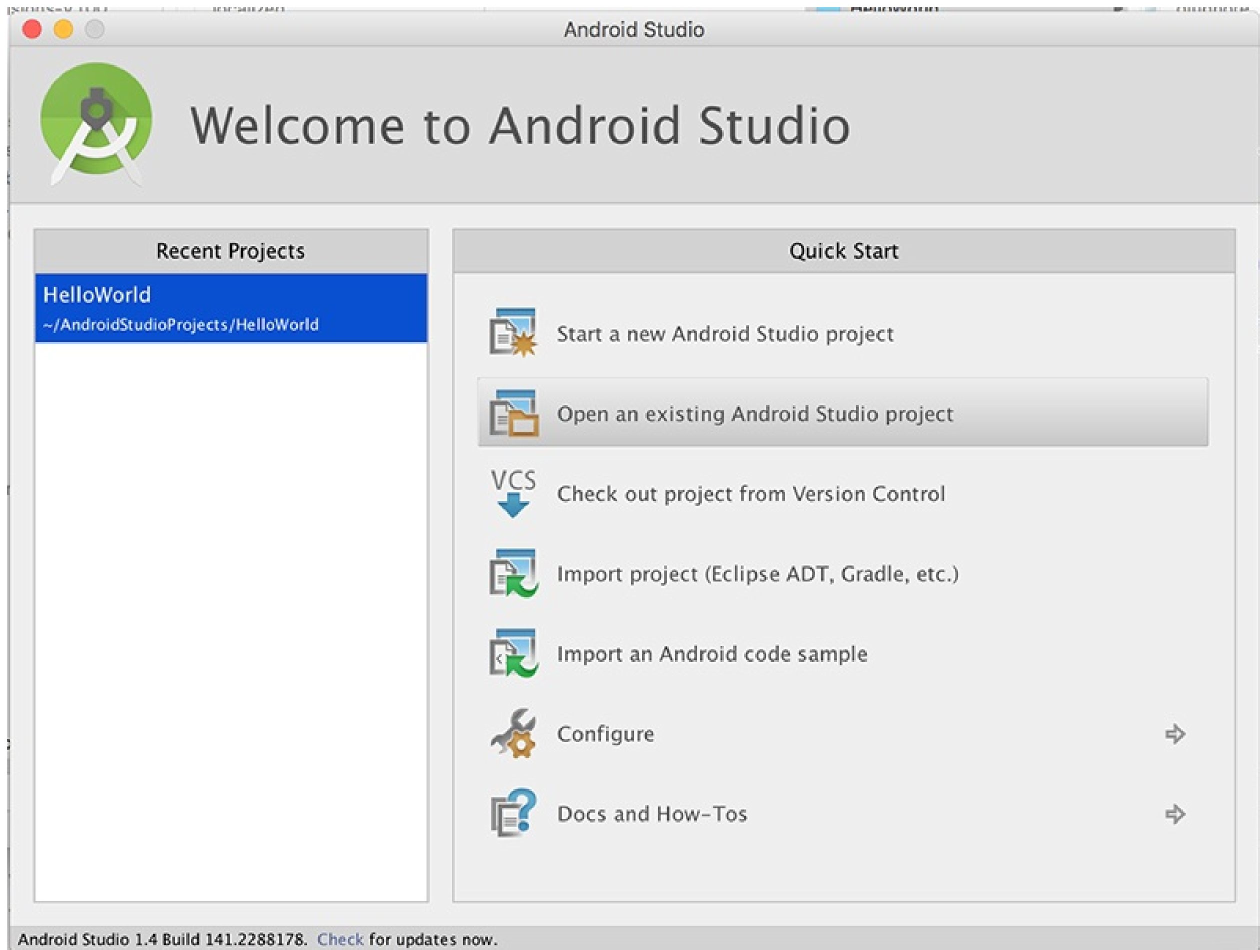


After you press "Finish", you should now have something similar to the following:

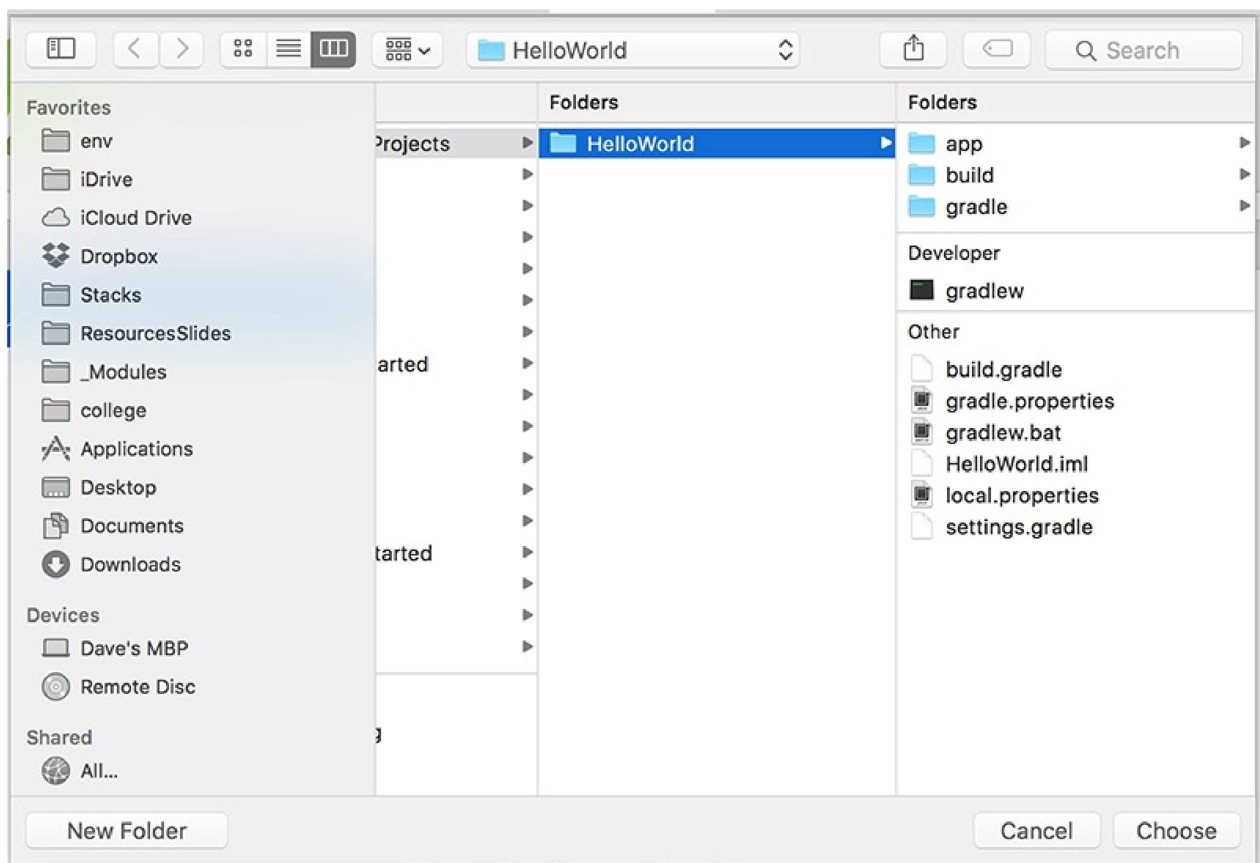


Next, as an exercise, select File->Close Project, to close the project so we can import it again.

If no other Projects are open you will be displayed with

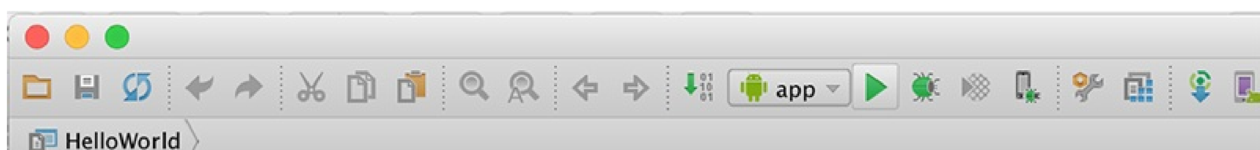


So, to import the project, select "Open an Existing Android Project" and navigate to the Project folder where you android app is stored (like **'HelloWorld'** below)

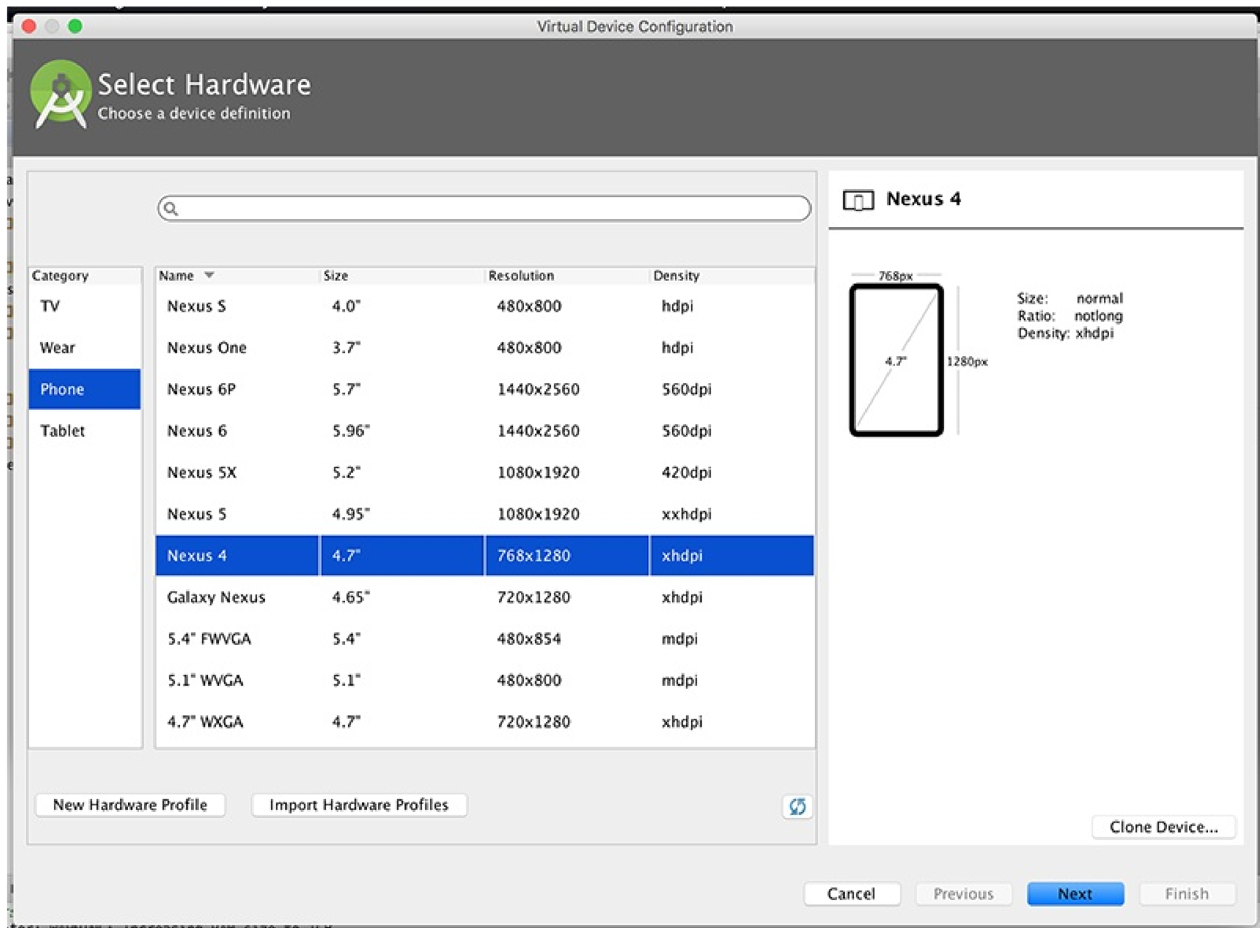


Once the project is open again, familiarise yourself with the project layout - the initial xml layout or "screen" is first displayed, this is one of the many resources you will be using and creating throughout this module. We will experiment later with modifying this layout, but first you should run the application.

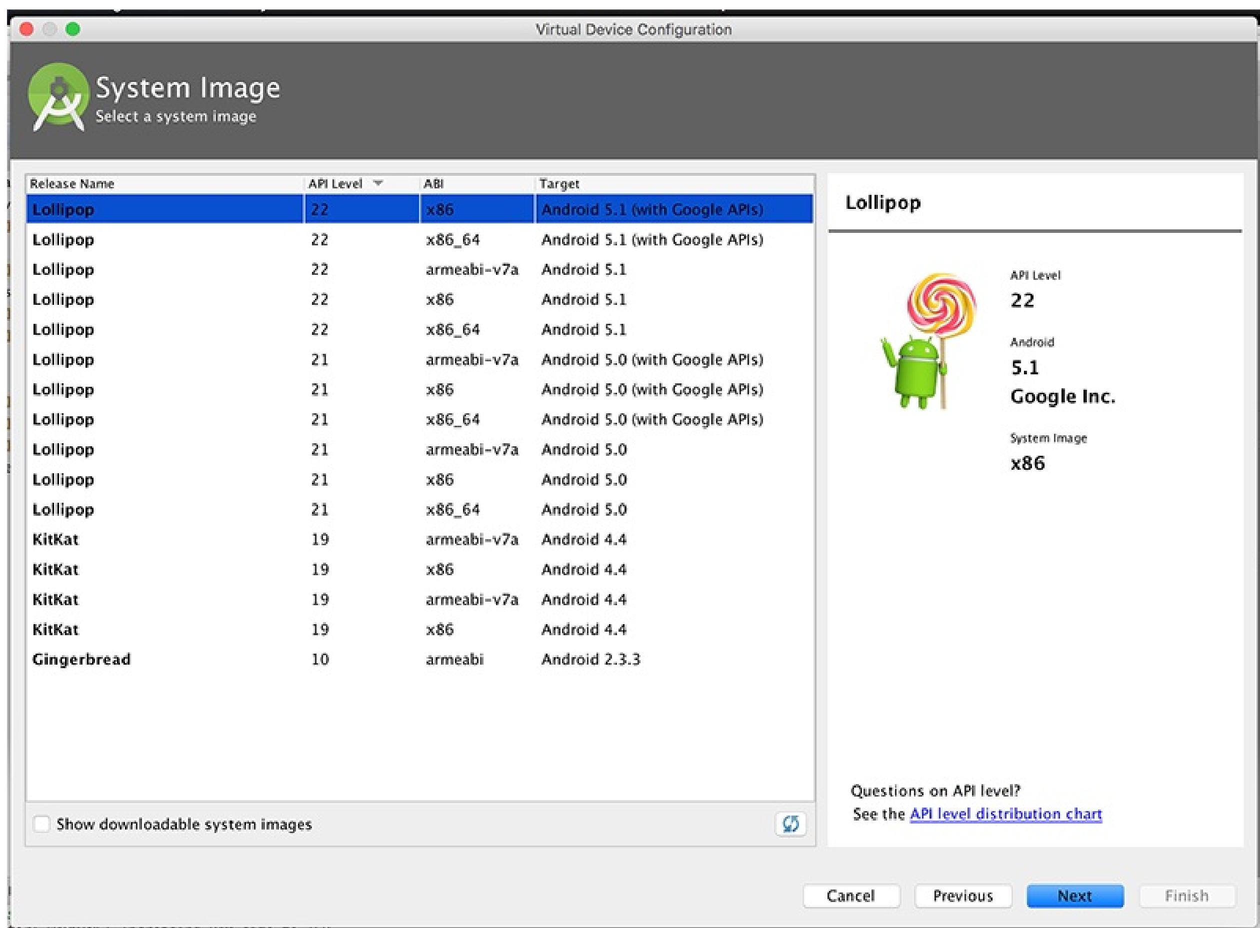
Select the Project (HelloWorld) and then select the 'Play' button as below



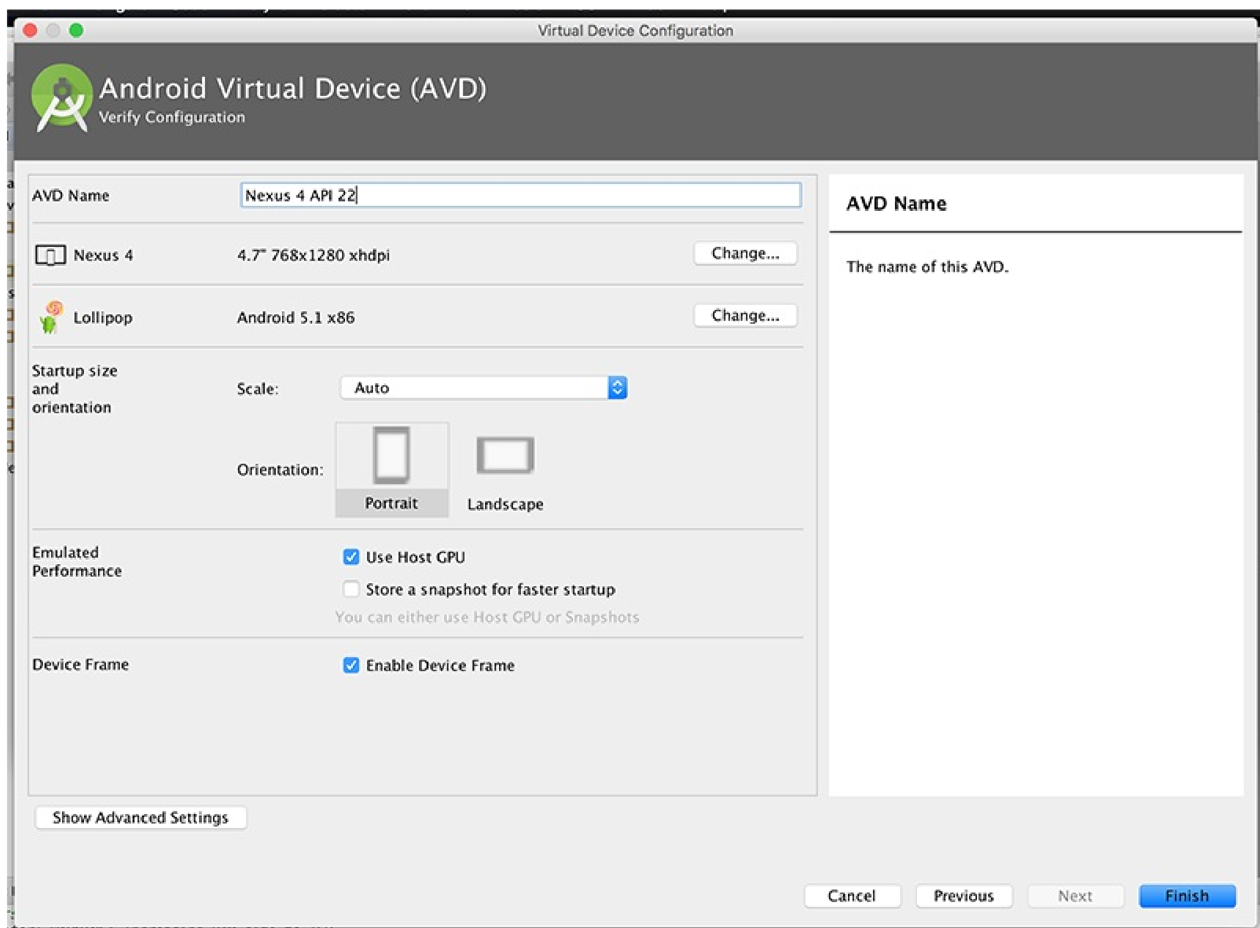
If you haven't done so already, you will be asked to select/create an AVD (Android Virtual Device), as follows:


Virtual Device Configuration
Select Hardware
Choose a device definition
Nexus 4
Category
TV
Wear
Phone
Tablet
Name
Size
Resolution
Density
Nexus S 4.0" 480x800 hdpi
Nexus One 3.7" 480x800 hdpi
Nexus 6P 5.7" 1440x2560 560dpi
Nexus 6 5.96" 1440x2560 560dpi
Nexus 5X 5.2" 1080x1920 420dpi
Nexus 5 4.95" 1080x1920 xxhdpi
Nexus 4 4.7" 768x1280 xhdpi
Galaxy Nexus 4.65" 720x1280 xhdpi
5.4" FWVGA 5.4" 480x854 mdpi
5.1" WVGA 5.1" 480x800 mdpi
4.7" WXGA 4.7" 720x1280 xhdpi
768px
4.7"
1280px
Size: normal
Ratio: notlong
Density: xhdpi
New Hardware Profile
Import Hardware Profiles
Clone Device...
Cancel
Previous
Next
Finish

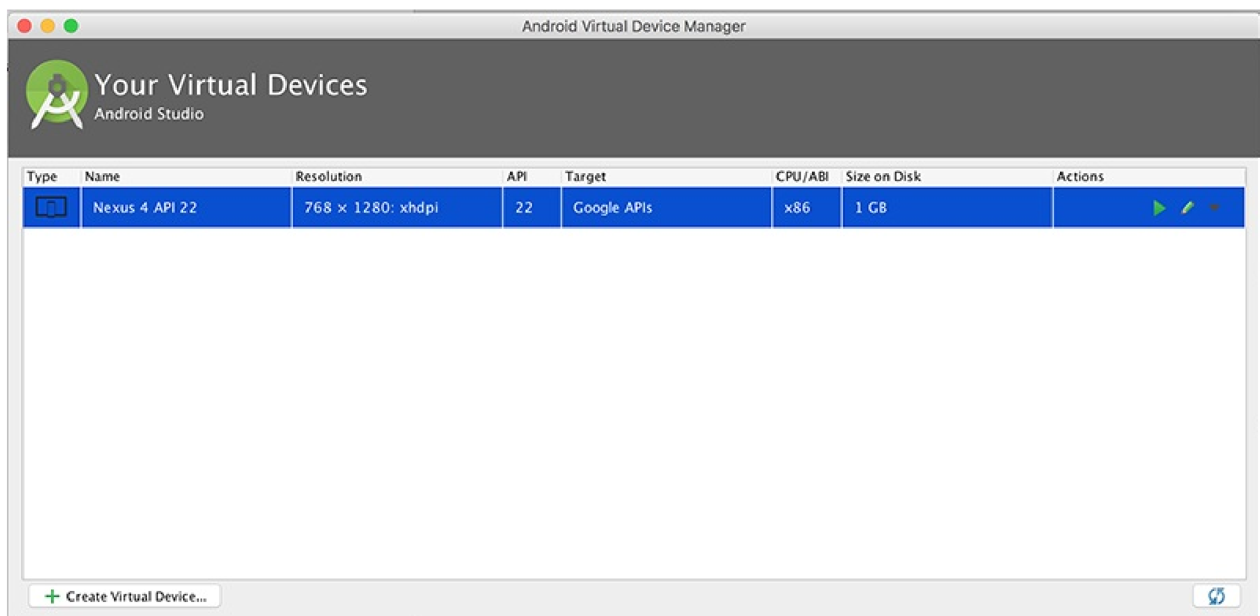
and


Virtual Device Configuration
System Image
Select a system image
Release Name
API Level
ABI
Target
Lollipop 22 x86 Android 5.1 (with Google APIs)
Lollipop 22 x86_64 Android 5.1 (with Google APIs)
Lollipop 22 armeabi-v7a Android 5.1
Lollipop 22 x86 Android 5.1
Lollipop 22 x86_64 Android 5.1
Lollipop 21 armeabi-v7a Android 5.0 (with Google APIs)
Lollipop 21 x86 Android 5.0 (with Google APIs)
Lollipop 21 x86_64 Android 5.0 (with Google APIs)
Lollipop 21 armeabi-v7a Android 5.0
Lollipop 21 x86 Android 5.0
Lollipop 21 x86_64 Android 5.0
KitKat 19 armeabi-v7a Android 4.4
KitKat 19 x86 Android 4.4
KitKat 19 armeabi-v7a Android 4.4
KitKat 19 x86 Android 4.4
Gingerbread 10 armeabi Android 2.3.3
Show downloadable system images
Lollipop
API Level
22
Android
5.1
Google Inc.
System Image
x86
Questions on API level?
See the API level distribution chart
Cancel
Previous
Next
Finish

and



and



The Virtual Device is very heavy on resources so you may need to choose settings below what is selected in the Screenshots.

Otherwise, use the settings as above and your first Android App should launch, (Once you've unlocked the device!):

Our version of "HelloWorld"

In this Step, you will be required to develop and run your own version of the "Hello World" Android Project (as seen below).



If you've deleted your HelloWorld Project, launch Android Studio (if it's not already open) and create a new Android Project called **HelloWorld** similar to what you did in Step 02.

If you haven't deleted the project, you can just continue on.

Name your package 'ie.wit' (like you did before). Choose an Empty Activity again and rename as before. It's recommended you select **Android 5.0** as the launch target platform (but any target will suffice for this particular lab). It's also probably a good idea to run the App at this stage, so you can set up your Virtual Device (if you haven't done so already).

Edit your "strings.xml" file (in your res folder) and add the following "resources" - be careful if you have created an app which contains a 'menu' folder, this also includes associated resources, so don't overwrite those resources, just add our ones at the end.

```
    <string name="window_text">Press the button below to receive
 a friendly greeting from Android.</string>
    <string name="button_label">Show Greeting</string>
    <string name="greeting_text">Hello from Android!</string>
```

Your completed strings.xml (without menus) should look like this

```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="window_text">Press the button below to receive
 a friendly greeting from Android.</string>
    <string name="button_label">Show Greeting</string>
    <string name="greeting_text">Hello from Android!</string>
</resources>
```

If you choose "open editor you can see the graphical representation of the String resources you have set up (and edit them here if you need to).

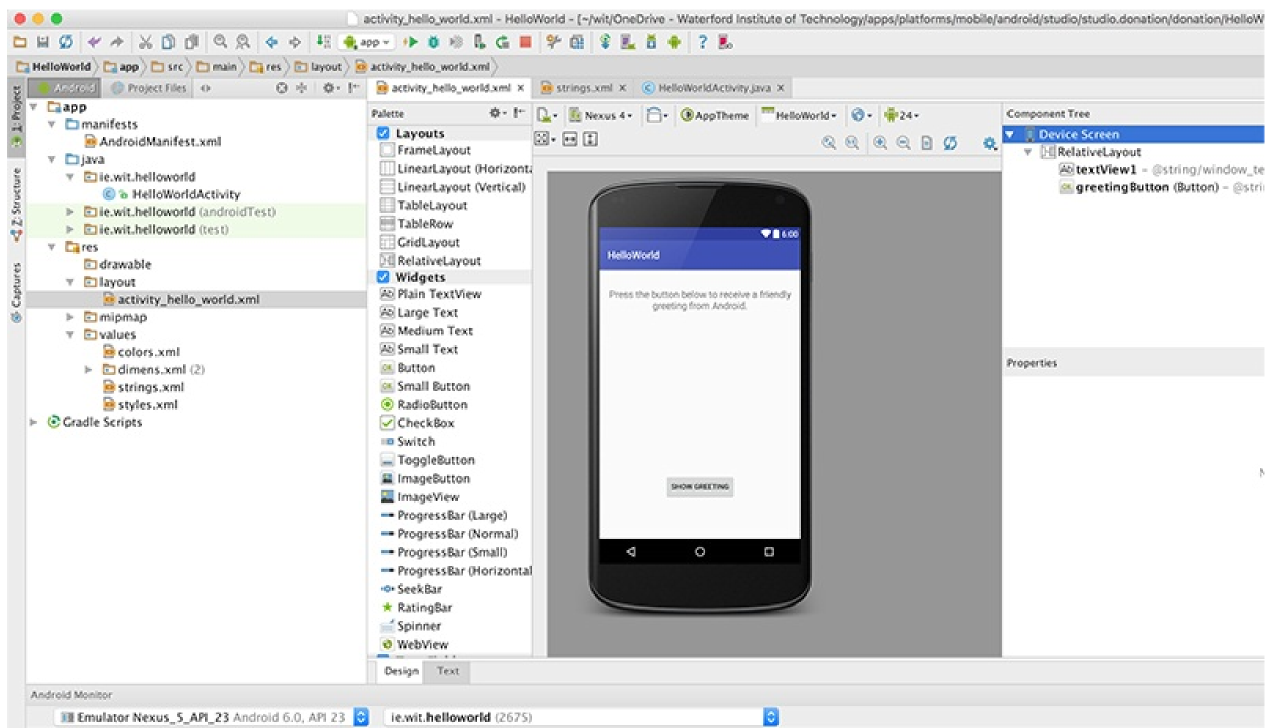


Now, edit your **"activity_hello_world.xml"** *in your **layout** folder* and replace your **TextView** with the following - make sure your in 'Text' view and not 'Design' view window.

```
<TextView
        android:id="@+id/textView1"
        android:gravity="center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="@string/window_text"
        android:textAppearance="?android:attr/textAppearanceMedi
um" />

    <Button
        android:id="@+id/greetingButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="59dp"
        android:onClick="showGreeting"
        android:text="@string/button_label" />
```

This will give you the following layout:



Once again, it's worth running the app at this point to confirm everything is displayed the way we want it. If you click the button, your app will probably crash - we don't have our **showGreeting()** method implemented yet.

So, the last thing we need to do is add in our event handling code so that a short message is displayed when the user presses the 'Show Greeting' button.

Firstly, open up the "HelloWorldActivity.java" source file and add the following method

```
public void showGreeting(View v) {
        // TODO Auto-generated method stub
        String greetingText = getString(R.string.greeting_text);
        Toast.makeText(this, greetingText, Toast.LENGTH_LONG).sh
ow();
    }
```

You'll get a few compiler errors due to missing imports, so try and fix those.

Note that we have no need for some kind of Listener interface (ala swing development) - our event handling is taken care of via the 'onClick' attribute in our xml layout, here's what your completed Activity class should look like.

```
package ie.wit.helloworld;

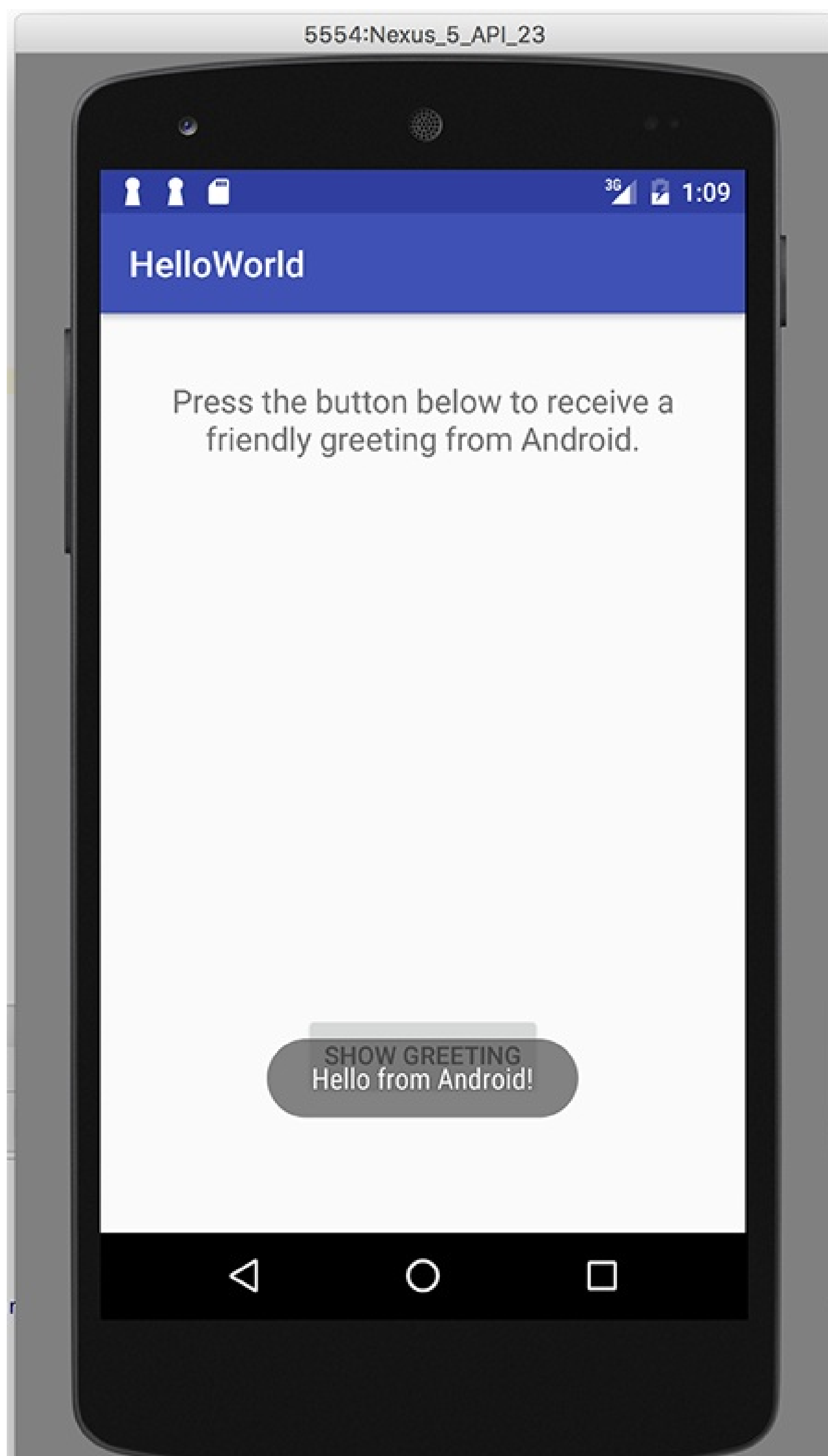
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Toast;

public class HelloWorldActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hello_world);
    }

    public void showGreeting(View v) {
        String greetingText = getString(R.string.greeting_text);
        Toast.makeText(this, greetingText, Toast.LENGTH_LONG).sh
ow();
    }
}
```

So when you run your app again you should see something like this when you click the 'Show Greeting' button.



We will investigate this code more closely in the lectures.

Exercises

Working with Resources

Just to get used to adding and editing resources, create a new button for our main layout, and try and 'hook it up' to a new string resource message to display to the user.

UI Design

Have a look at adding in a new colour resource and changing the default background colour for the layout.

Lab 02: Introduction to 'Donation'

This is our first look at the **Donation** case study, and will involve building an initial single screen app, with some common Android widgets on our layout.

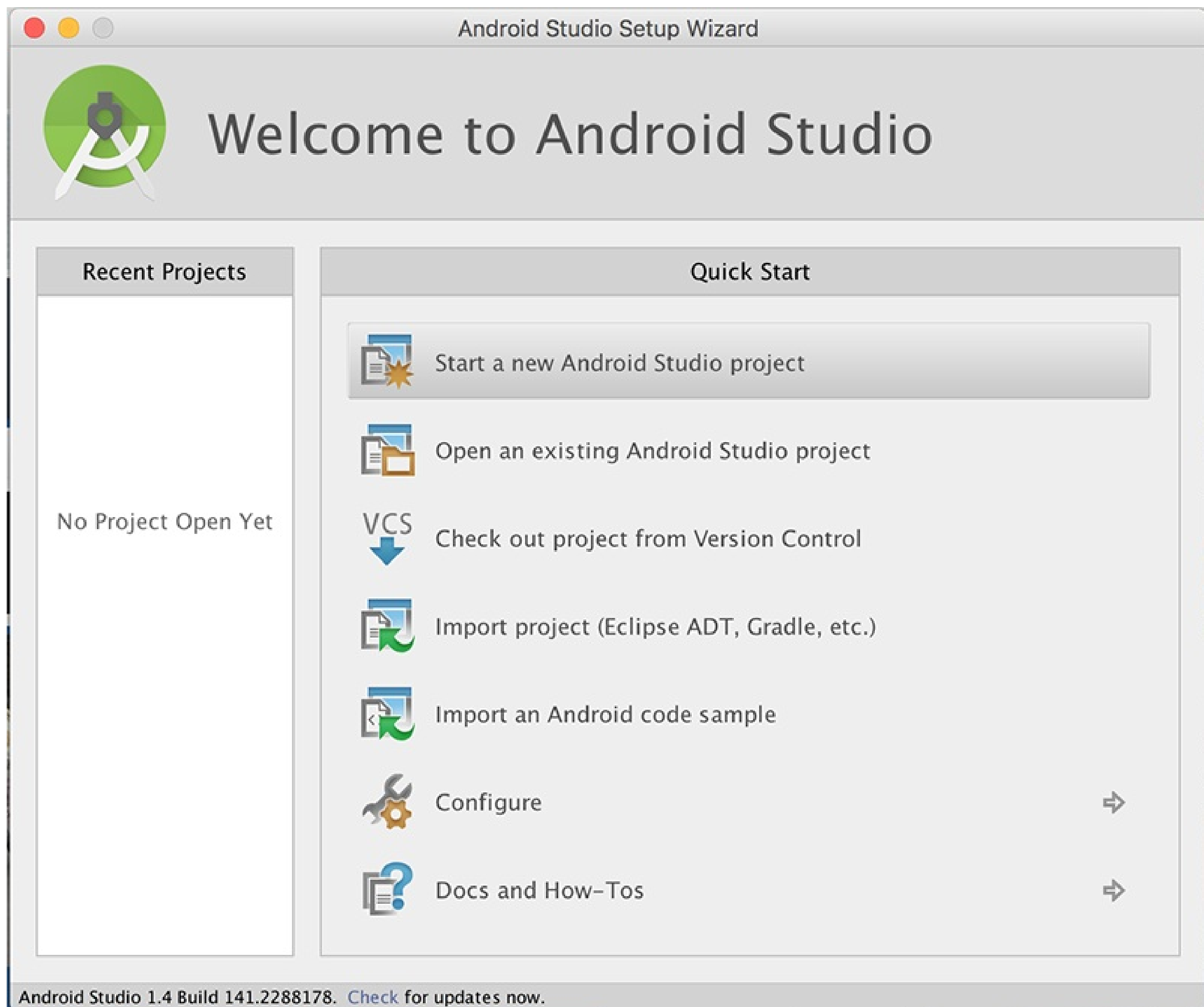
Objectives

In this Lab, you will be required to build the first version of our Donation Case Study App , called **Donation.1.0**. We will build on this lab (and the Case Study) over the following weeks and add in some new features and UI Design along the way. In this version we will add in a few UI widgets on a single layout and implement some basic event handling.
On completion of this lab you'll be able to

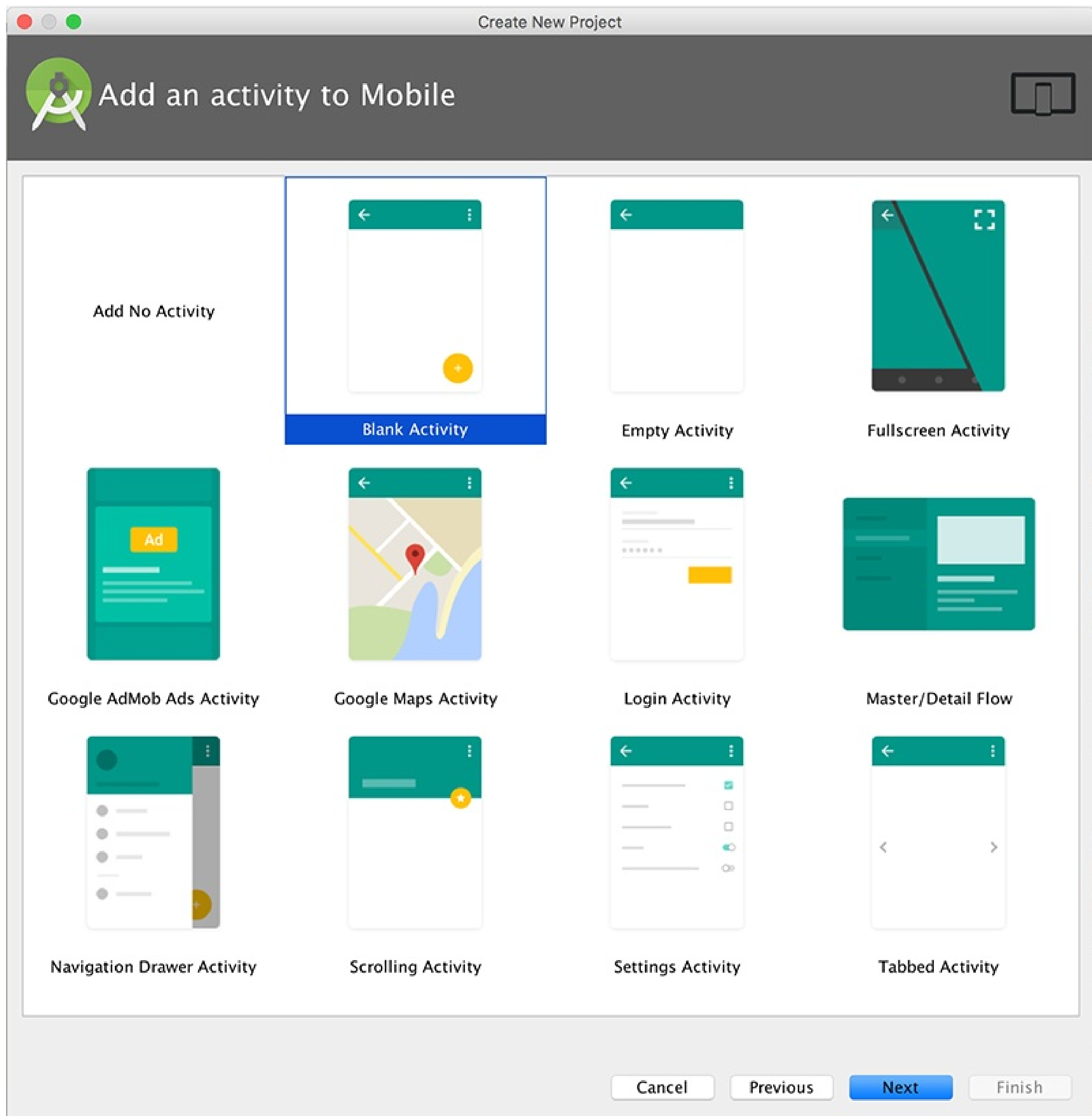
- design an initial layout for the app.
- add different widgets (such as a button, text and picker) to a layout.
- implement basic event handling to cause some action on the click of a button.
- be able to 'filter' Log messages for testing and debugging

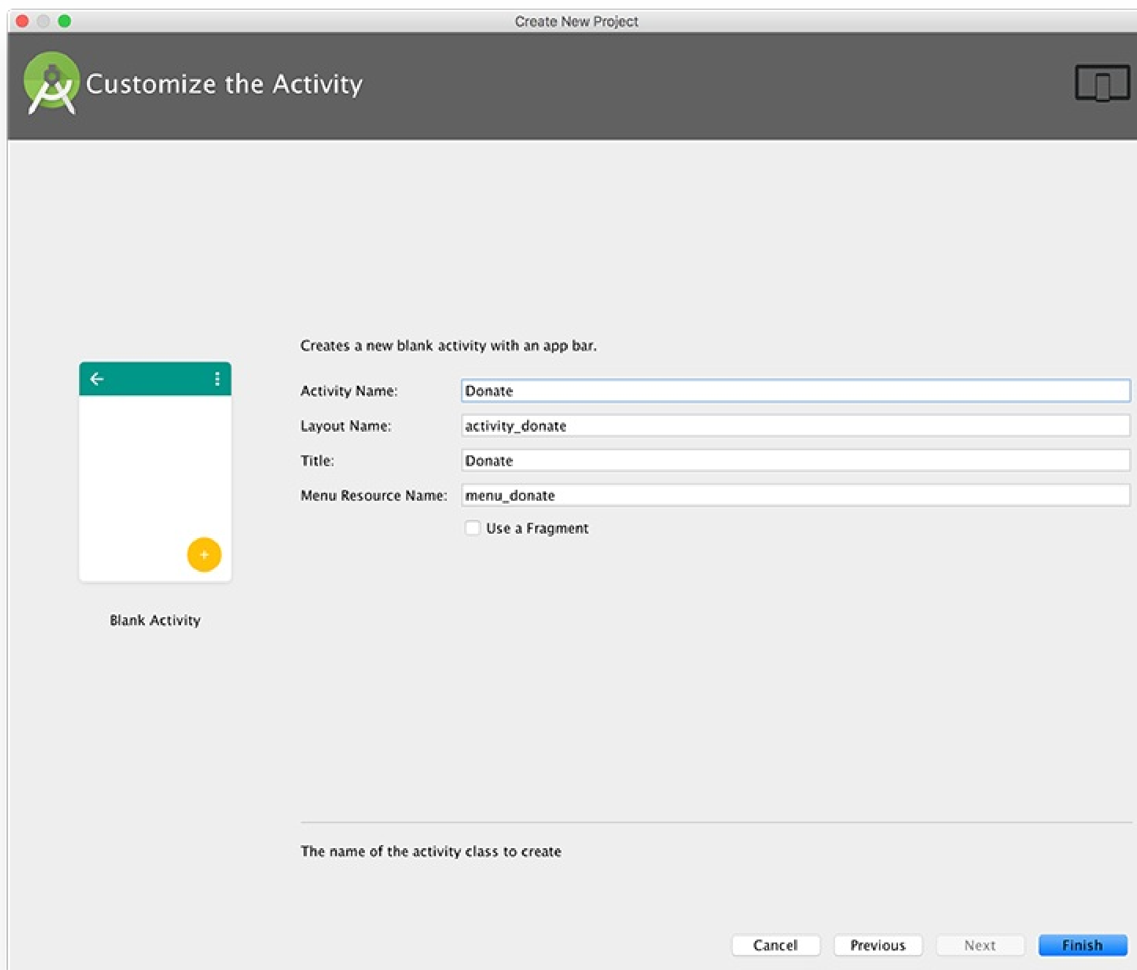
Step 01 - Create Project

Create a new Android application, as demonstrated in the following 5 screenshots:

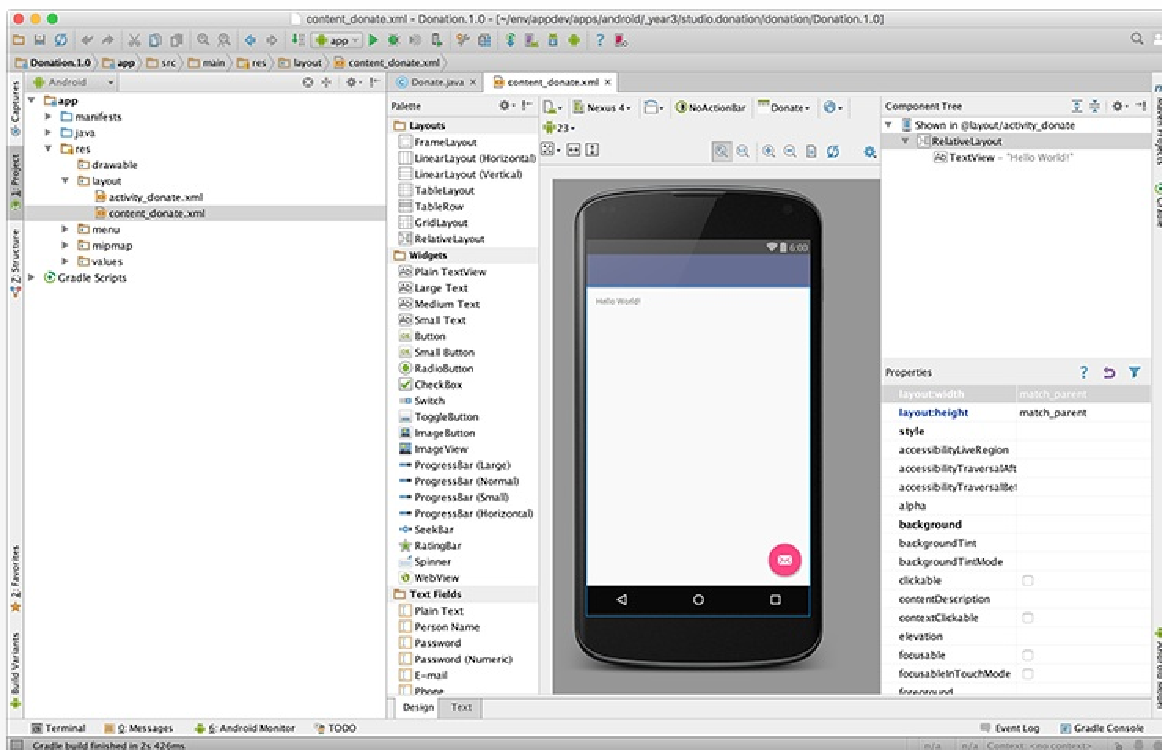


Note the 'Minimum Required SDK' which may be different from the default (as above).

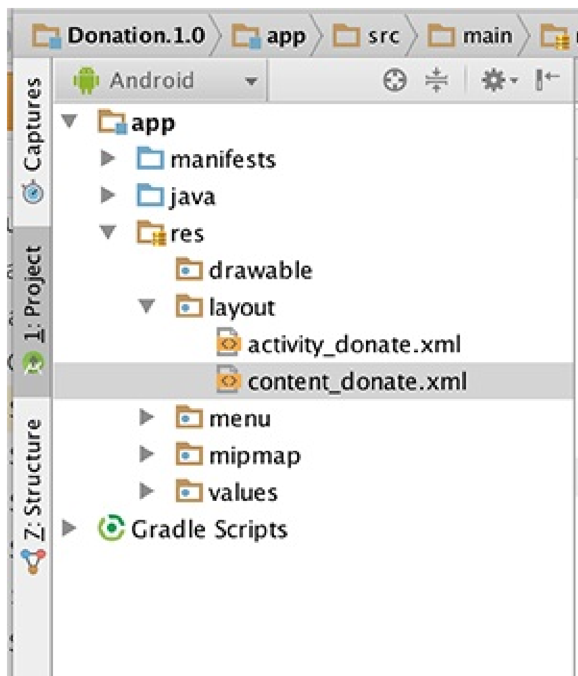

Create New Project
Add an activity to Mobile
Add No Activity
Blank Activity
Empty Activity
Fullscreen Activity
Ad
Google AdMob Ads Activity
Google Maps Activity
Login Activity
Master/Detail Flow
Navigation Drawer Activity
Scrolling Activity
Settings Activity
Tabbed Activity
Cancel
Previous
Next
Finish



The opening project perspective is shown below, with the content_donate layout open in the visual designer:

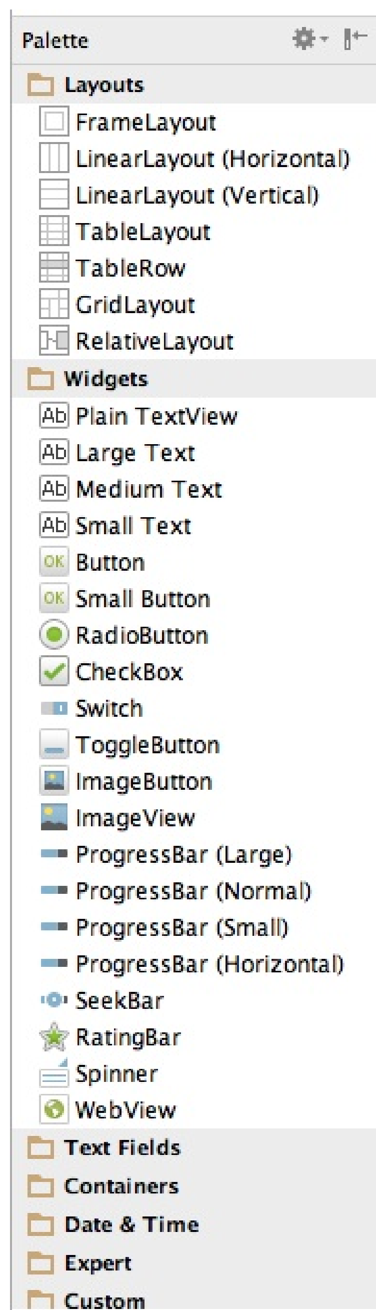


The project will look like this

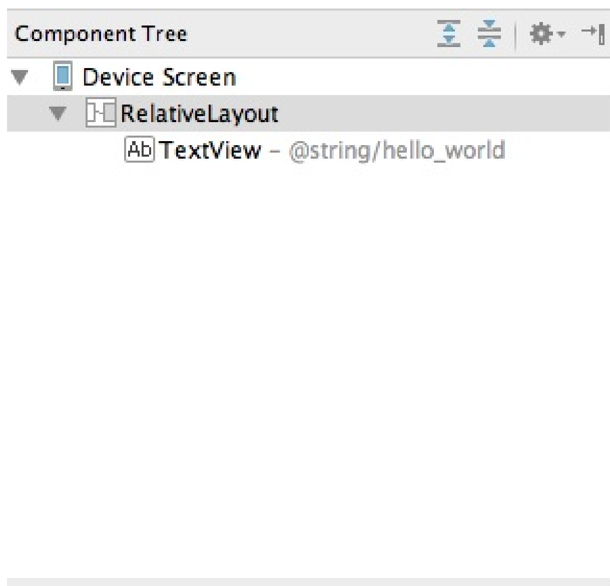


It is important to become familiar with the structure and purpose of the three panes surrounding the Donation 'canvas':

Pallette:


Palette
Layouts
FrameLayout
LinearLayout (Horizontal)
LinearLayout (Vertical)
TableLayout
TableRow
GridLayout
RelativeLayout
Widgets
Plain TextView
Large Text
Medium Text
Small Text
Button
Small Button
RadioButton
CheckBox
Switch
ToggleButton
ImageButton
ImageView
ProgressBar (Large)
ProgressBar (Normal)
ProgressBar (Small)
ProgressBar (Horizontal)
SeekBar
RatingBar
Spinner
WebView
Text Fields
Containers
Date & Time
Expert
Custom

Outline



Properties

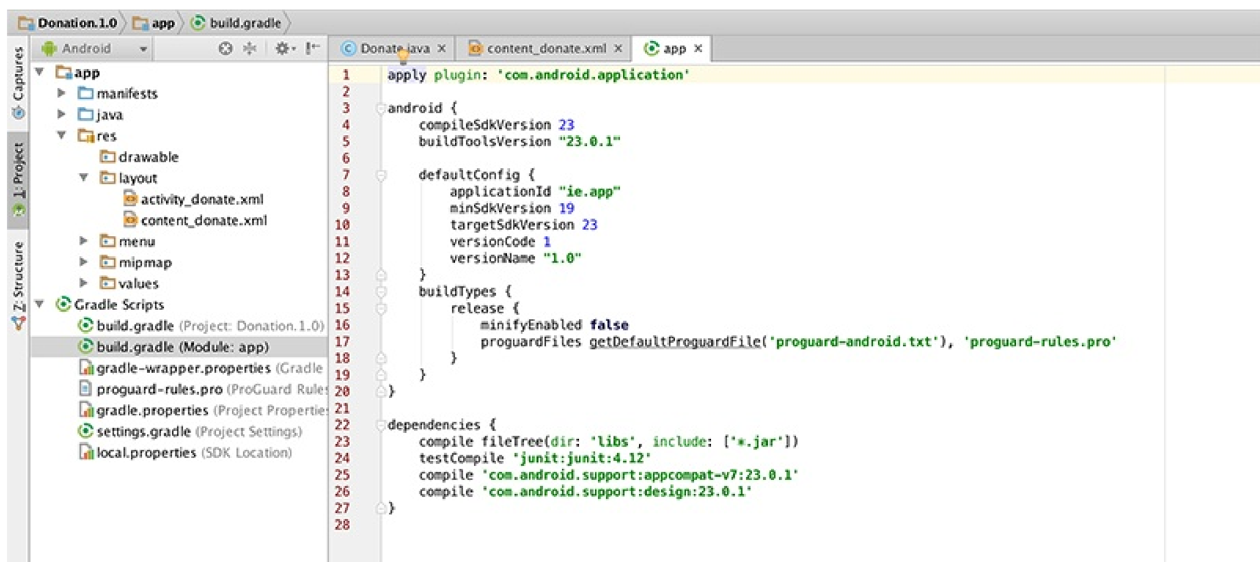
Properties

layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveRegion	
alpha	
background	
clickable	☐
contentDescription	
focusable	☐
focusableInTouchMode	☐
gravity	[]
id	
ignoreGravity	
importantForAccessibilit	
labelFor	
layoutMode	

These views are closely related - and you will need to monitor the information displayed there continually as you evolve the appearance of your activities screens.

Also, take a quick look at your **build.gradle** file to see the configuration and dependencies of your app.

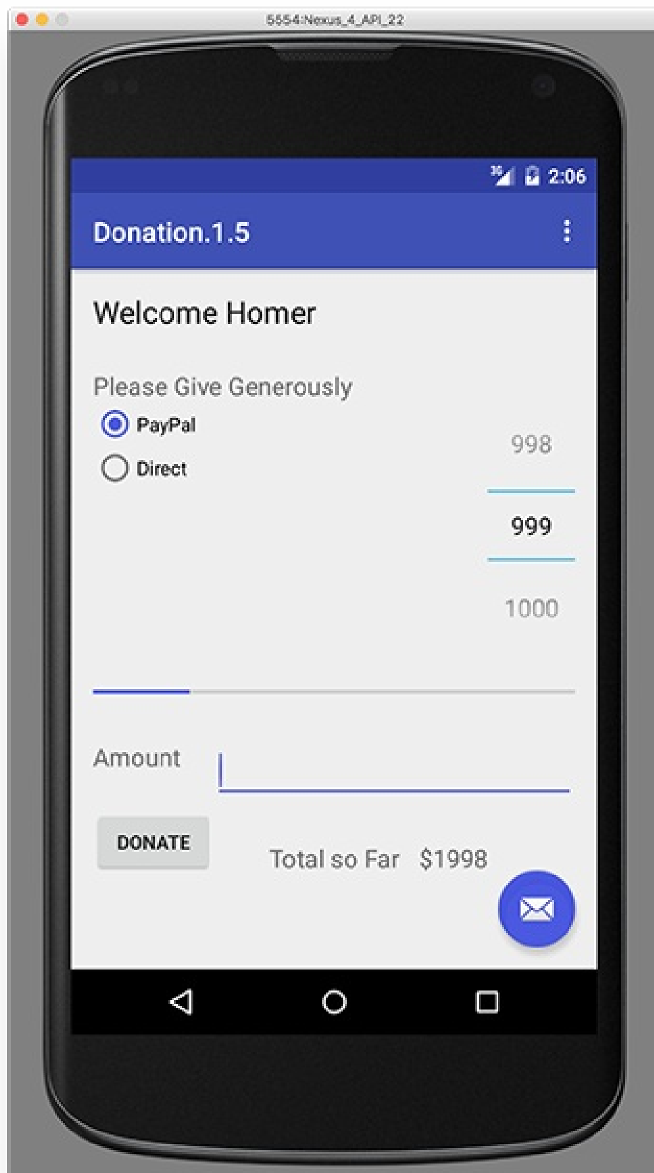
Step 01



Donation.1.0
app
build.gradle
Android
Donate.java
content_donate.xml
app
manifests
java
res
drawable
layout
activity_donate.xml
content_donate.xml
menu
mipmap
values
Gradle Scripts
build.gradle (Project: Donation.1.0)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle
proguard-rules.pro (ProGuard Rules
gradle.properties (Project Propertie
settings.gradle (Project Settings)
local.properties (SDK Location)
apply plugin: 'com.android.application'
android {
compileSdkVersion 23
buildToolsVersion "23.0.1"
defaultConfig {
applicationId "ie.app"
minSdkVersion 19
targetSdkVersion 23
versionCode 1
versionName "1.0"
}
buildTypes {
release {
minifyEnabled false
proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
}
}
}
dependencies {
compile fileTree(dir: 'libs', include: ['*.jar'])
testCompile 'junit:junit:4.12'
compile 'com.android.support:appcompat-v7:23.0.1'
compile 'com.android.support:design:23.0.1'
}

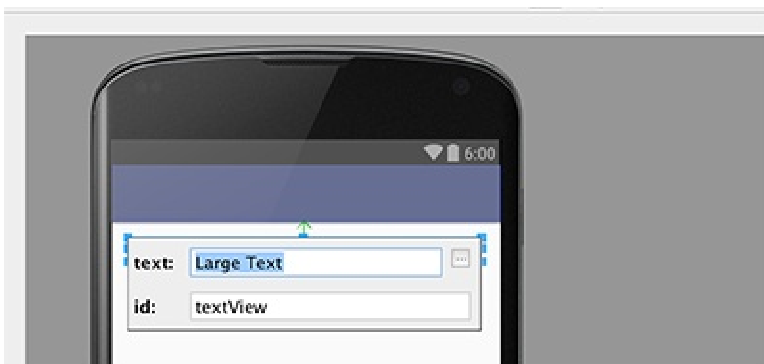
Step 02 - Layout Donation Activity

For this lab, our objective is to produce an Android App that looks something like this:

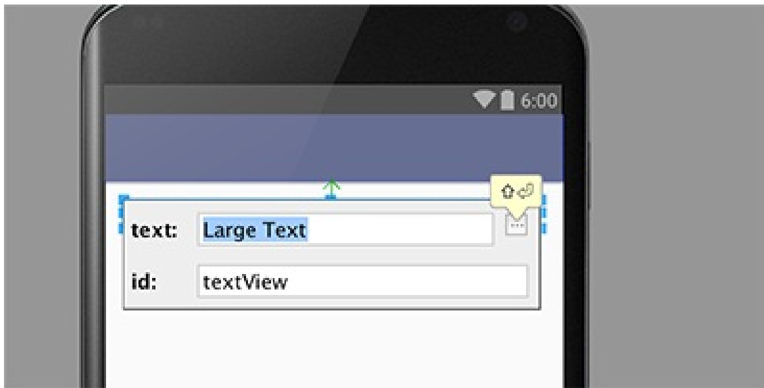


In your **content_donate.xml**, delete the current 'Hello World' text, and drag and drop a new 'LargeText' form widget onto the canvas, and 'stretch' the widget to fill the canvas (like below). Look closely at the following:

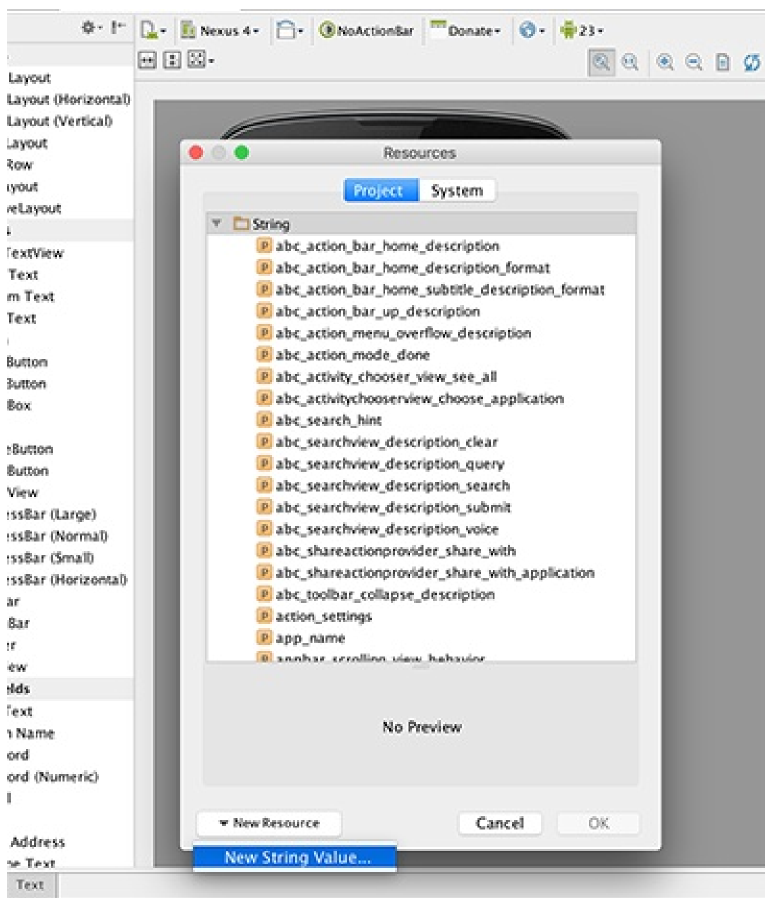
Now, Double-Click the widget and you will be presented with the following:



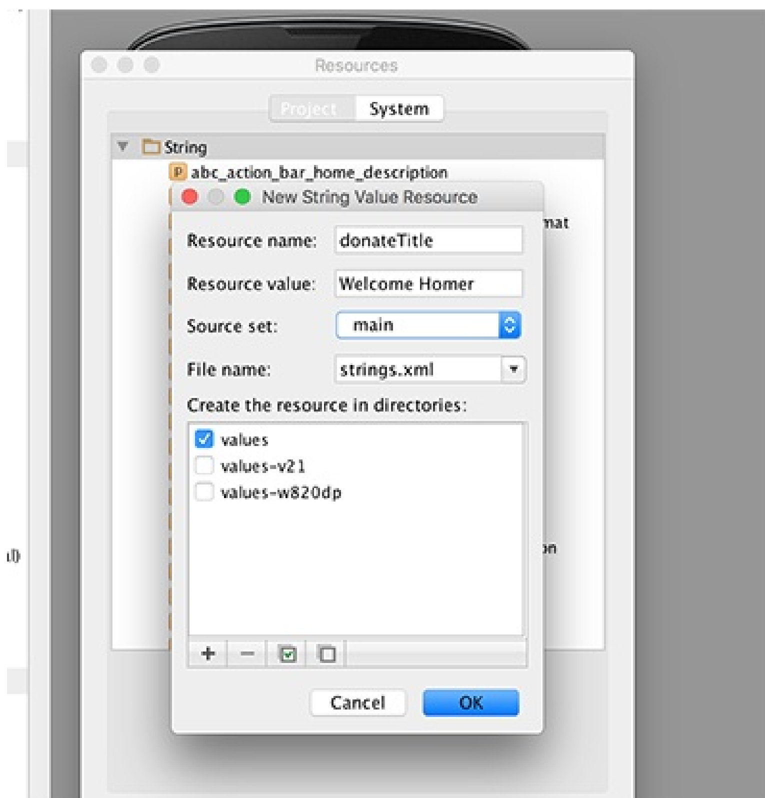
Select the elipse (on the right hand side, indicated below)



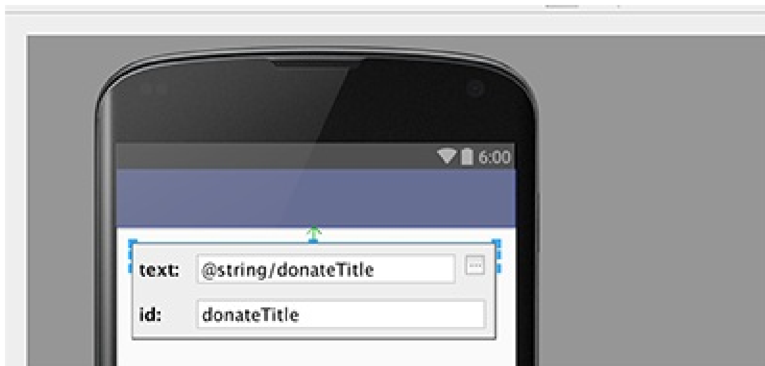
and you will be presented with the Resources Menu



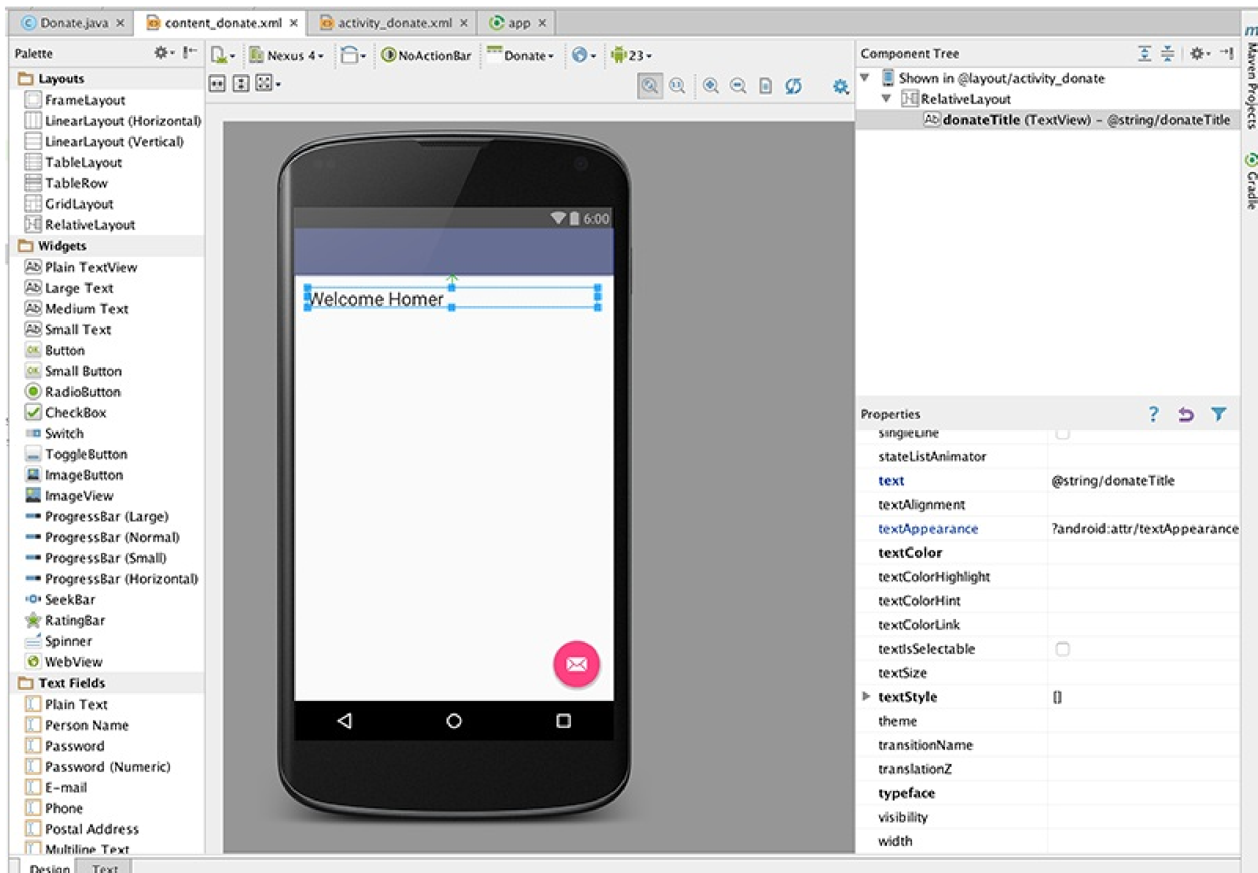
Select a 'New Resource->New String Value, and fill in the values as below



Double-Click the widget again and enter **donteTitle** for the **id** and hit return.

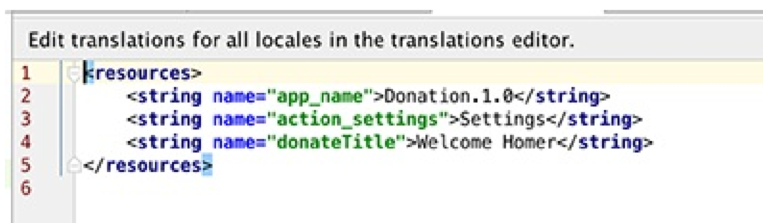


Once completed, you'll have something like this



Note carefully the following features:

- the guides tyeing the text to the left, top and right corner
- in Outline - the name of the control has been changed from a default to 'donateTitle'.
- in Properties - check the value for 'text' and notice we have a string reference linking to our **strings.xml** (below)



Edit translations for all locales in the translations editor.

```
1 <resources>
2     <string name="app_name">Donation.1.0</string>
3     <string name="action_settings">Settings</string>
4     <string name="donateTitle">Welcome Homer</string>
5 </resources>
6
```

Locate the following two files and inspect them closely:

res/layout/content_dontate.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/re
s/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto" android:
layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="@d
imen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_donate" tools:context=".Donat
e">

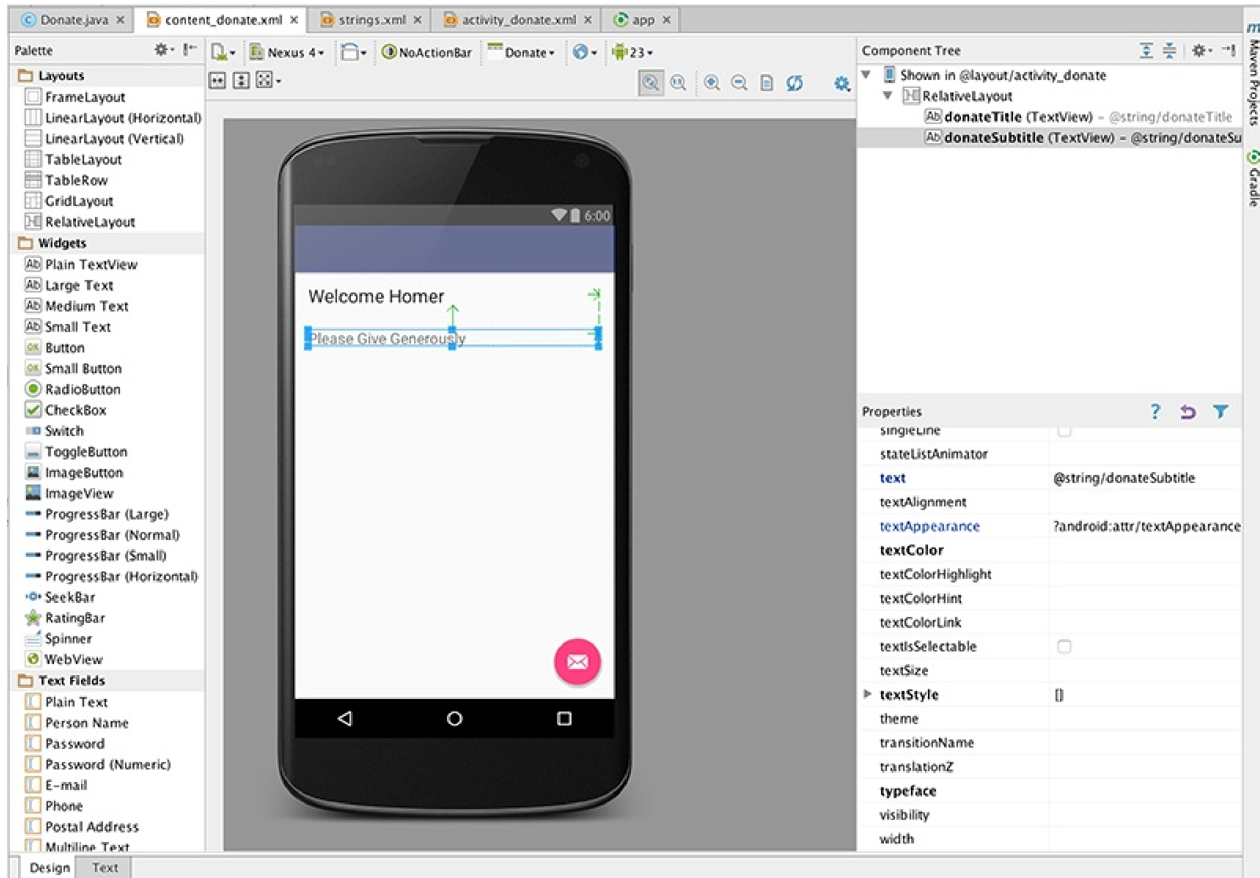
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarg
e"
        android:text="@string/donateTitle"
        android:id="@+id/donateTitle"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true" />
</RelativeLayout>
```

res/values/strings.xml

```
<resources>
    <string name="app_name">Donation</string>
    <string name="action_settings">Settings</string>
    <string name="donateTitle">Welcome Homer</string>
</resources>
```

Note the relationship between 'donateTitle' in both files.

Bring in the following string into the donate activity now - (medium text) - and follow the same procedure as above. The designer should look like this:



and our XML files will look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/re
s/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto" android:
layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="@d
imen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_donate" tools:context=".Donat
e">

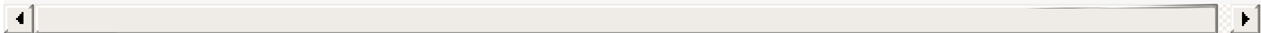
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarg
e"
        android:text="@string/donateTitle"
        android:id="@+id/donateTitle"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedi
um"
        android:text="@string/donateSubtitle"
        android:id="@+id/donateSubtitle"
        android:layout_below="@+id/donateTitle"
        android:layout_alignParentStart="true"
        android:layout_marginTop="27dp"
        android:layout_alignEnd="@+id/donateTitle" />
</RelativeLayout>
```

Our 'strings.xml' file....

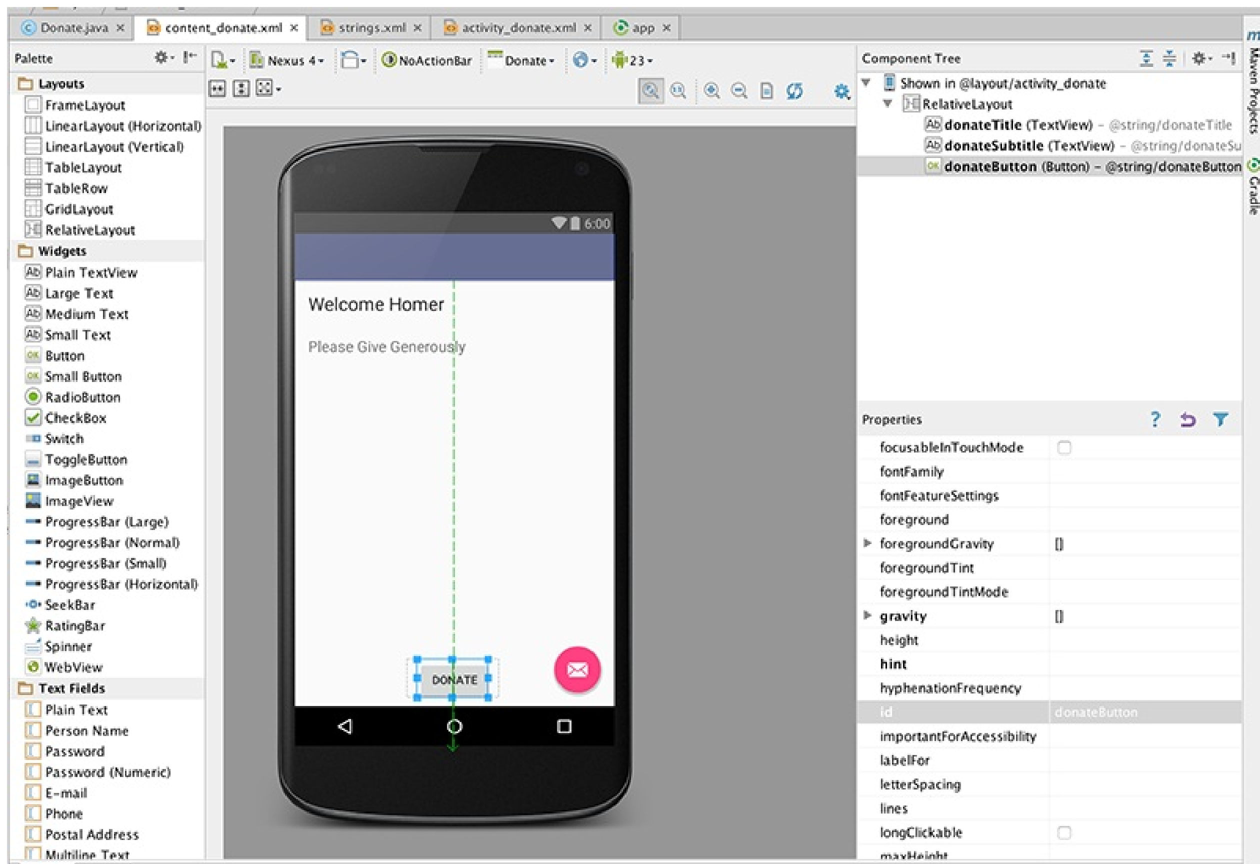
```
<resources>
    <string name="app_name">Donation.1.0</string>
    <string name="action_settings">Settings</string>
    <string name="donateTitle">Welcome Homer</string>
    <string name="donateSubtitle">Please Give Generously</string>

</resources>
```



Step 03 - The 'Donate' Button

Place a button directly on to the activity - attached to the bottom of the screen as shown:



Following a similar procedure as in the previous step, rename the button and add an id, both called **donateButton** If all goes as expected, your xml files should be like this:

activity_donate.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/re
s/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto" android:
layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="@d
```

```
imen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_donate" tools:context=".Donat
e">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarg
e"
        android:text="@string/donateTitle"
        android:id="@+id/donateTitle"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedi
um"
        android:text="@string/donateSubtitle"
        android:id="@+id/donateSubtitle"
        android:layout_below="@+id/donateTitle"
        android:layout_alignParentStart="true"
        android:layout_marginTop="27dp"
        android:layout_alignEnd="@+id/donateTitle" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/donateButton"
        android:id="@+id/donateButton"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="47dp" />
</RelativeLayout>
```

strings.xml

```
<resources>
    <string name="app_name">Donation.1.0</string>
    <string name="action_settings">Settings</string>
    <string name="donateTitle">Welcome Homer</string>
    <string name="donateSubtitle">Please Give Generously</string>

    <string name="donateButton">Donate</string>
</resources>
```

If there is a deviation from the above - retrace your steps (delete the button) until you can match the above.

We can now switch our attention to the Java Activity class Donate:

```
package ie.app;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;

public class Donate extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_donate);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findVi
```

```
ewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own actio
n", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar i
f it is present.
        getMenuInflater().inflate(R.menu.menu_donate, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar wi
ll
        // automatically handle clicks on the Home/Up button, so
 long
        // as you specify a parent activity in AndroidManifest.x
ml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

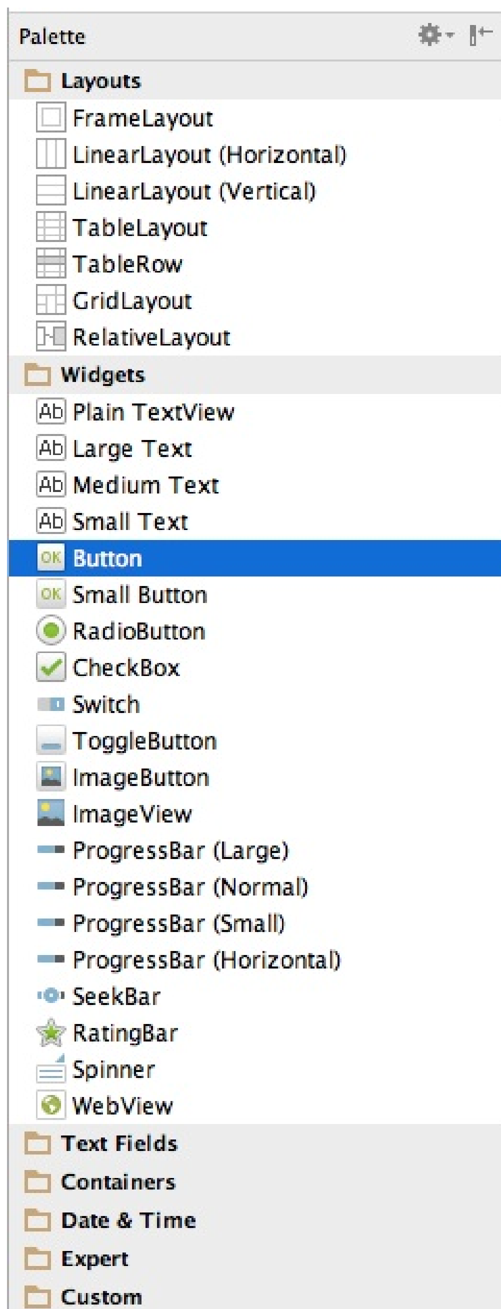
        return super.onOptionsItemSelected(item);
    }
}
```

For any 'controls' a user can interact with we usually find it useful to associate a class member with that object. Currently we only have one - a Button. The text fields we don't consider 'interactive' as such, so we will not include those.

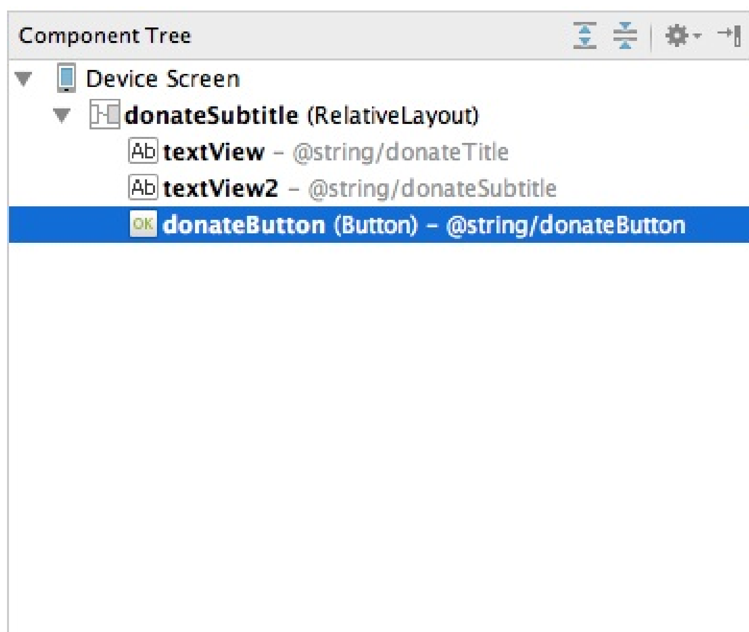
Insert the following new field into the class:

```
  private Button donateButton;
```

The class will have to be imported. The class name will always match the name in the Palette:



We are free to call the variable anything we like. However, in order to keep confusion to a minimum, always call the variable by the same name you used in the Outline view:



In onCreate - we need to initialise this variable:

```
    donateButton = (Button) findViewById(R.id.donateButton);
```

We might also add a logging message so we can have some feedback as the app launches:

```
    if (donateButton != null)
        {
            Log.v("Donate", "Really got the donate button");
        }
```

This is the complete activity class:

```
package ie.app;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
```

```
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;

public class Donate extends AppCompatActivity {

    private Button donateButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_donate);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findVi
ewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own actio
n", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });

        donateButton = (Button) findViewById(R.id.donateButton);

        if (donateButton != null)
        {
            Log.v("Donate", "Really got the donate button");
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar i
```

```
f it is present.
        getMenuInflater().inflate(R.menu.menu_donate, menu);
        return true;
    }

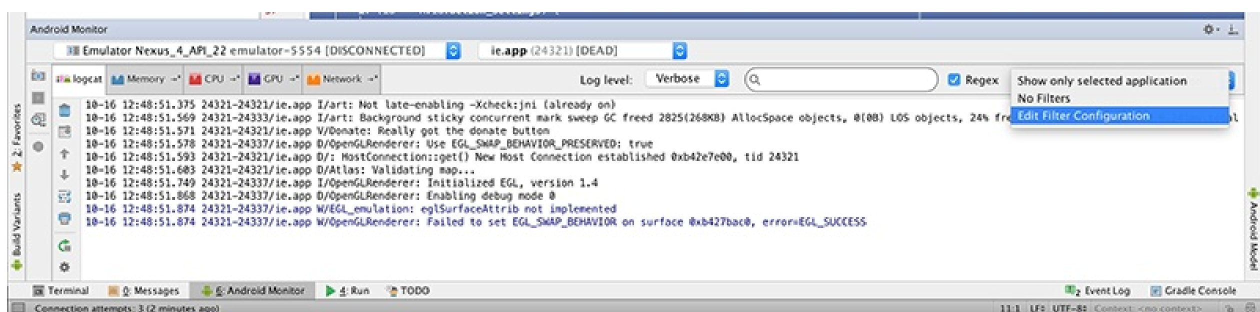
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar wi
ll
        // automatically handle clicks on the Home/Up button, so
 long
        // as you specify a parent activity in AndroidManifest.x
ml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

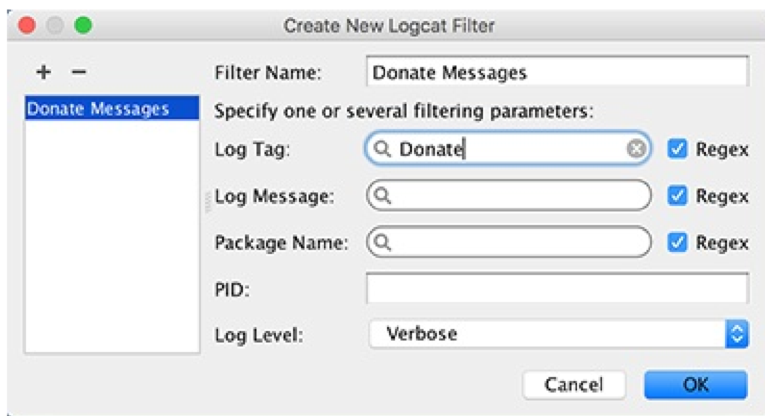
        return super.onOptionsItemSelected(item);
    }
}
```

Finding the log message can be very difficult, unless you set a filter. In the 'LogCat' view in Android Studio, create a filter like this:

Choose "Edit Filter Configuration" on the right-hand-side of the LogCat View



Enter the name and filter as below



and if you then select the filter, we should see our message:



Run the app again, and verify the above message appears.

Step 04 - Documentation

The android documentation is particularly helpful and well designed. These are the two key starting points:

- http://developer.android.com/guide/components/index.html
- http://developer.android.com/reference/packages.html

The first is designed to be read though as a guide, perhaps independent of any work in Android Studio. You should get into the habit of devoting an hour or two a week just reading this section.

The Reference guide should always be open as you are working on labs or projects, and you should make a serious effort to get to grips with at least some of the information here.

Taking the Button class we have just started using. We can immediately find the reference just by knowing the import statement in our Activity class:

```
import android.widget.Button;
```

.. translates to

- http://developer.android.com/reference/android/widget/Button.html

(note the last three segments match the package name). Open this page now. Read just as far as the "Button Style" heading. There seems to be two ways of learning when an button event occurs. The first method is using the event handler/listener - but a second easier method is also available.

Try this now. Bring in a new method into Donate class:

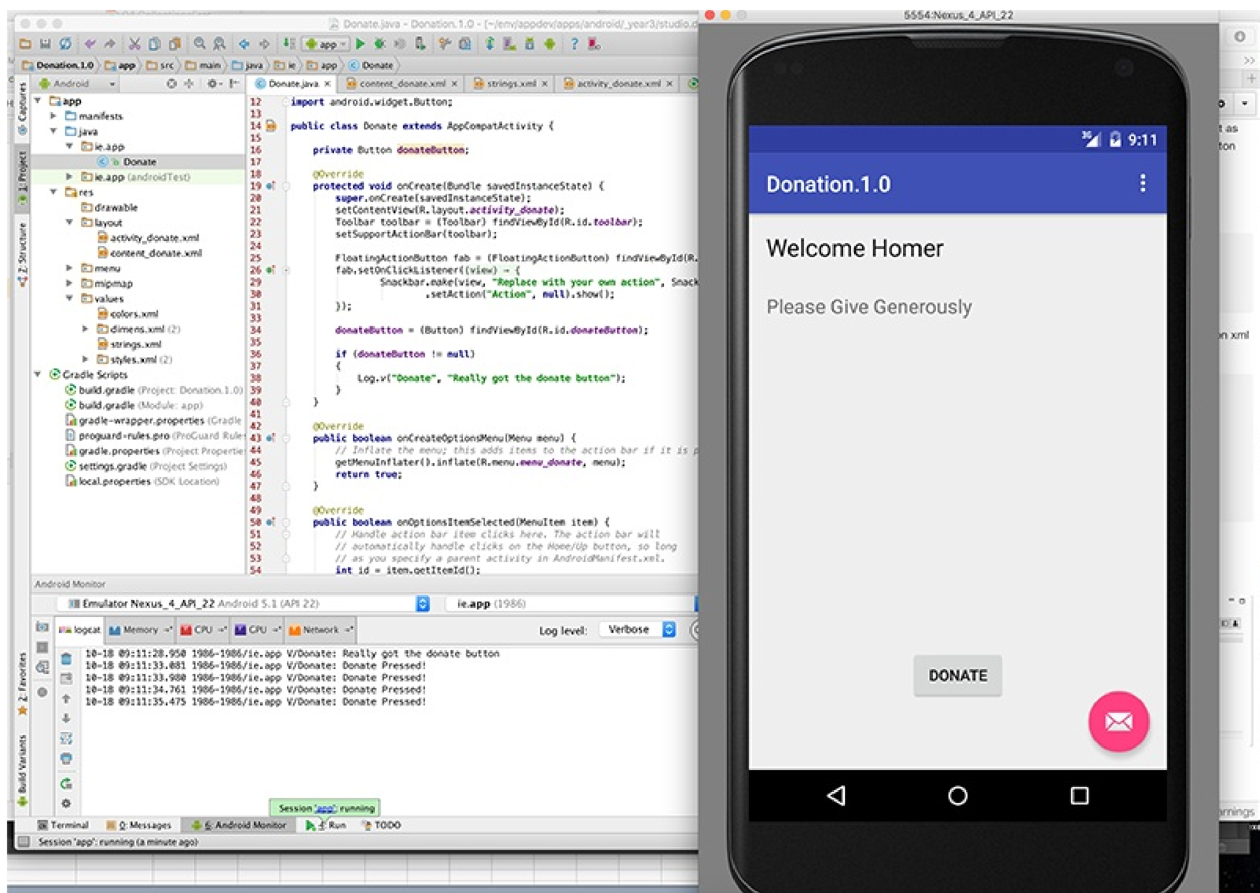
```
  public void donateButtonPressed (View view)
  {
    Log.v("Donate", "Donate Pressed!");
   }
```

Then, edit the **content_donate.xml** file **-** and add a new 'onClick' attribute into the Button xml fragment:

(the very last entry)

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/donateButton"
    android:id="@+id/donateButton"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="47dp"
    android:onClick="donateButtonPressed"/>
```

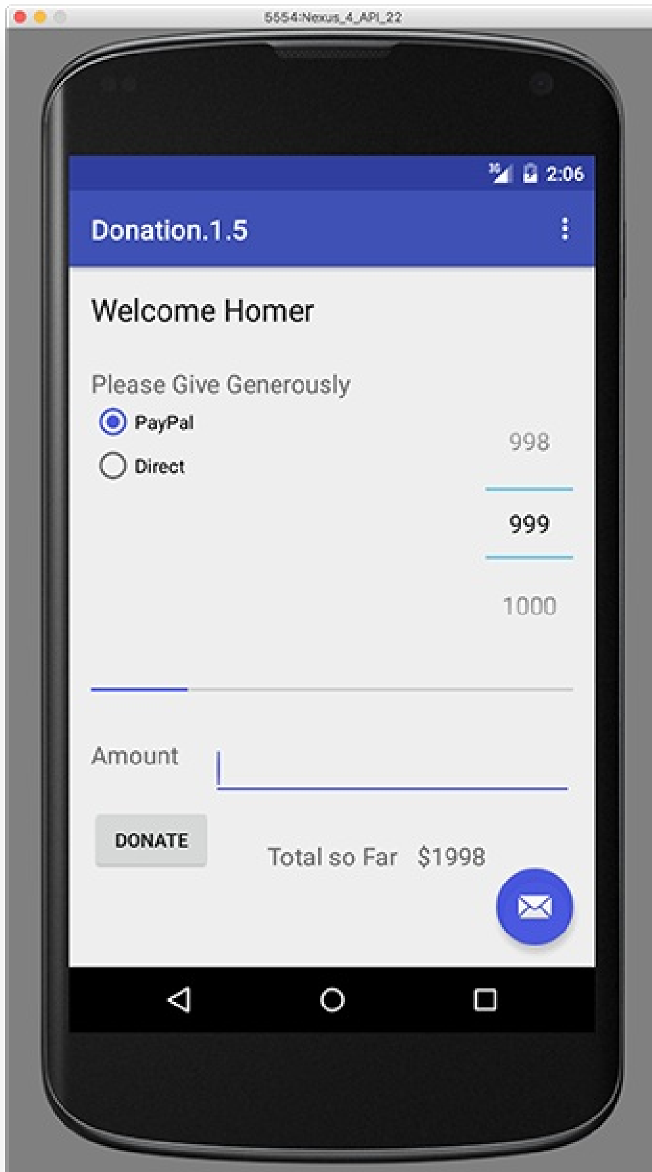
Save everything and execute the app, and monitor the log as you press the button:



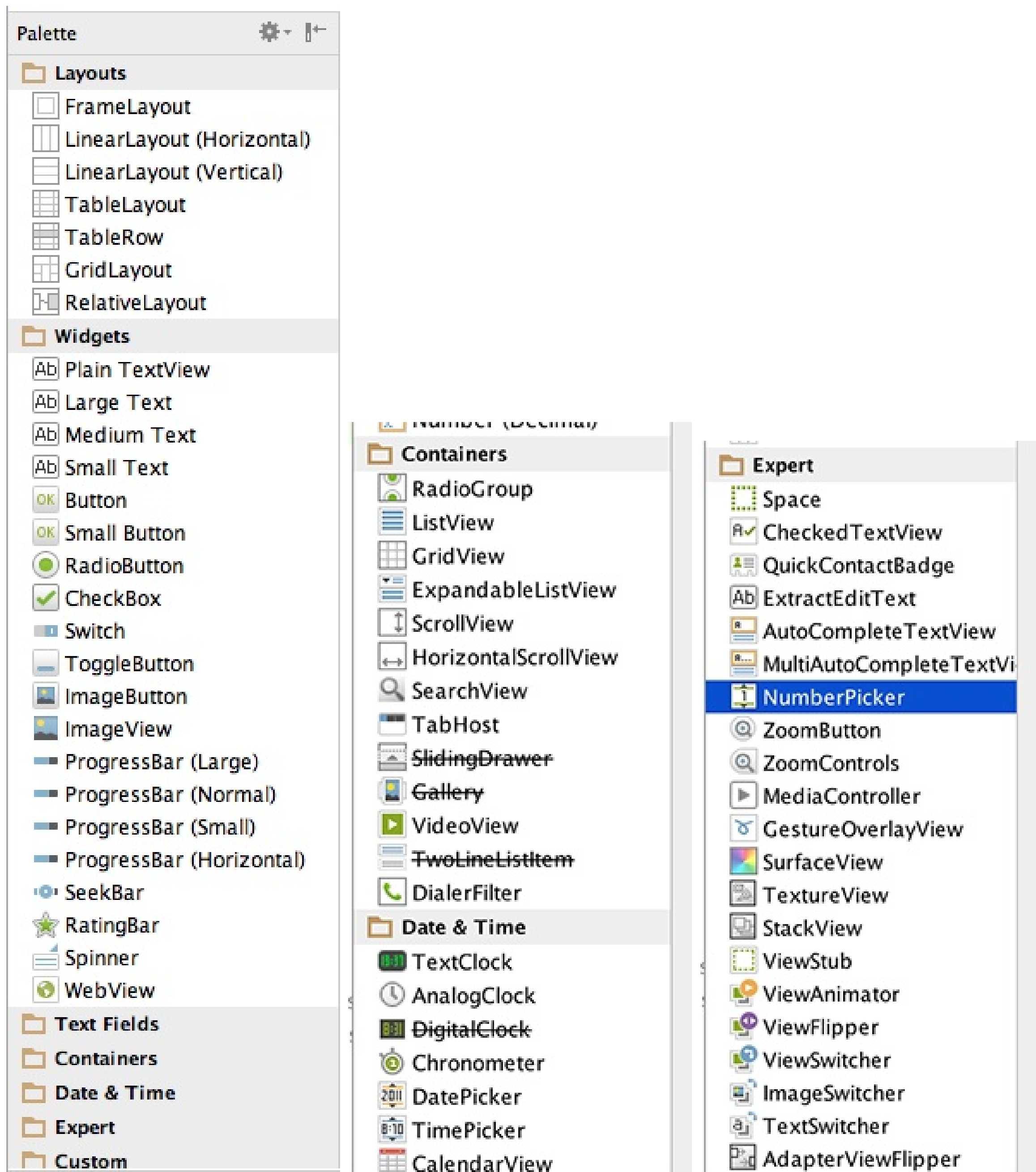
We now have our first interaction working!

Step 05 - New Control Layout

Recall the UI we are trying to implement:



We need radio buttons, some sort of selection/combo box + a progress bar. These can be found in various locations in the pallette:



RadioGroup, ProgressBar and NumberPicker seem likely candidates. The names of these controls are exactly as advertised, and we can expect them to be in the 'widgets' package. To verify this, try importing them at the top of the Donate activity class:

```
import android.widget.RadioGroup;
import android.widget.NumberPicker;
import android.widget.ProgressBar;
```

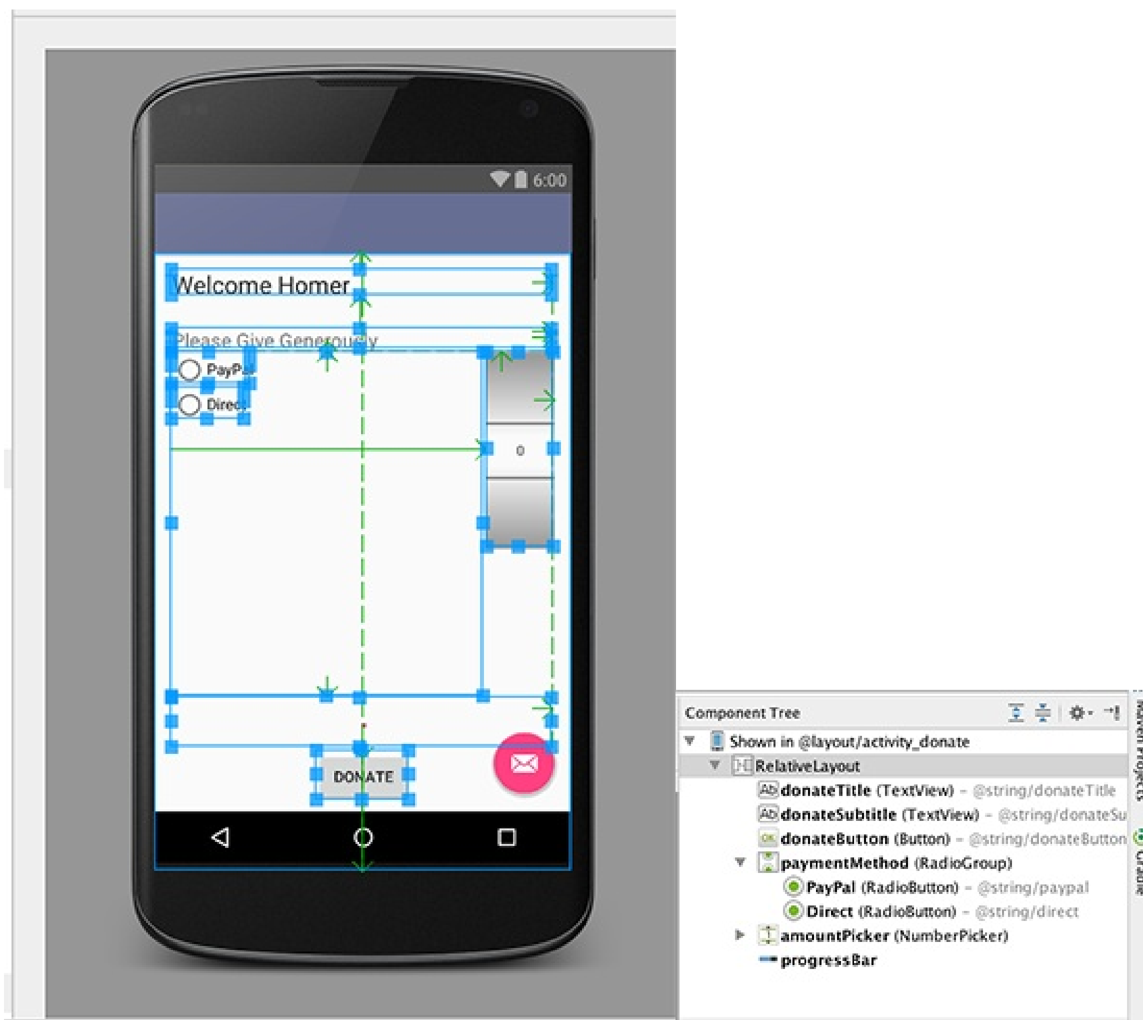
... and we can bring in three fields into the class:

```
private RadioGroup   paymentMethod;
private ProgressBar  progressBar;
private NumberPicker amountPicker;
```

We can also open up three pages of documentation - which we can reverse engineer from the package/class names:

- http://developer.android.com/reference/android/widget/RadioGroup.html
- http://developer.android.com/reference/android/widget/ProgressBar.html
- http://developer.android.com/reference/android/widget/NumberPicker.html

Note this time we have gone to the Activity class before actually creating the controls. We should do this now - and remember to use the same names (for the IDs) as we create the controls.



Getting the layout +id names as shown above may take some practice. However, it is an essential skill to get on top of, even it it takes a lot of trial and error.

For reference purposes (try to do it yourself first!), these are the relevant generated xml files:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/re
s/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto" android:
layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="@d
imen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_donate" tools:context=".Donat
e">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarg
e"
        android:text="@string/donateTitle"
        android:id="@+id/donateTitle"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedi
um"
        android:text="@string/donateSubtitle"
        android:id="@+id/donateSubtitle"
        android:layout_below="@+id/donateTitle"
        android:layout_alignParentStart="true"
        android:layout_marginTop="27dp"
        android:layout_alignEnd="@+id/donateTitle" />
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/donateButton"
    android:id="@+id/donateButton"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="47dp"
    android:onClick="donateButtonPressed"/>

<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_below="@+id/donateSubtitle"
    android:layout_alignParentStart="true"
    android:id="@+id/paymentMethod"
    android:layout_toStartOf="@+id/amountPicker"
    android:layout_above="@+id/progressBar">

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/paypal"
        android:id="@+id/PayPal"
        android:checked="false" />

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/direct"
        android:id="@+id/Direct"
        android:checked="false" />
</RadioGroup>

<NumberPicker
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/amountPicker"
    android:layout_alignTop="@+id/paymentGroup"
```

```
        android:layout_alignEnd="@+id/donateSubtitle" />

    <ProgressBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="?android:attr/progressBarStyleHorizontal"
        android:id="@+id/progressBar"
        android:layout_alignParentStart="true"
        android:layout_alignEnd="@+id/donateSubtitle"
        android:layout_above="@+id/donateButton"
        android:indeterminate="false" />
</RelativeLayout>
```

```
<resources>
    <string name="app_name">Donation.1.0</string>
    <string name="action_settings">Settings</string>
    <string name="donateTitle">Welcome Homer</string>
    <string name="donateSubtitle">Please Give Generously</string>

    <string name="donateButton">Donate</string>
    <string name="paypal">PayPal</string>
    <string name="direct">Direct</string>
</resources>
```

If we have our naming conventions right - then we can bind to these new controls in onCreate:

```
    paymentMethod = (RadioGroup)   findViewById(R.id.paymentMeth
od);
    progressBar   = (ProgressBar)  findViewById(R.id.progressBar
);
    amountPicker  = (NumberPicker) findViewById(R.id.amountPicke
r);
```

This is the complete Donate class so far:

```
package ie.app;
```

```
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.NumberPicker;
import android.widget.ProgressBar;
import android.widget.RadioGroup;

public class Donate extends AppCompatActivity {

    private Button          donateButton;
    private RadioGroup      paymentMethod;
    private ProgressBar     progressBar;
    private NumberPicker    amountPicker;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_donate);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findVi
ewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own actio
n", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
```

```
        donateButton = (Button) findViewById(R.id.donateButton);

        if (donateButton != null)
        {
            Log.v("Donate", "Really got the donate button");
        }

        paymentMethod = (RadioGroup)   findViewById(R.id.payment
Method);
        progressBar   = (ProgressBar)  findViewById(R.id.progres
sBar);
        amountPicker  = (NumberPicker) findViewById(R.id.amountP
icker);

        amountPicker.setMinValue(0);
        amountPicker.setMaxValue(1000);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar i
f it is present.
        getMenuInflater().inflate(R.menu.menu_donate, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar wi
ll
        // automatically handle clicks on the Home/Up button, so
 long
        // as you specify a parent activity in AndroidManifest.x
ml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }
```

```
        return super.onOptionsItemSelected(item);
    }

    public void donateButtonPressed (View view)
    {
        Log.v("Donate", "Donate Pressed!");
    }
}
```

Step 06 - NumberPicker

This is our reference documentation:

- http://developer.android.com/reference/android/widget/NumberPicker.html

which is a little overwhelming. Back in the guides:

- http://developer.android.com/guide/components/index.html

we might find some useful tutorial type introduction to this control - under 'User Interface' - 'Input Controls'

- http://developer.android.com/guide/topics/ui/controls.html

.. and this is the page on 'pickers'

- http://developer.android.com/guide/topics/ui/controls/pickers.html

This documentation is concerned with Fragments - a concept that may be difficult to grasp initially, and also explores the usage of date and time pickers.

We can get up and running without this much fuss. Returning to the documentation, these three methods should be sufficient initially:

- http://developer.android.com/reference/android/widget/NumberPicker.html#setMaxValue(int)
- http://developer.android.com/reference/android/widget/NumberPicker.html#setMinValue(int)
- http://developer.android.com/reference/android/widget/NumberPicker.html#getValue()

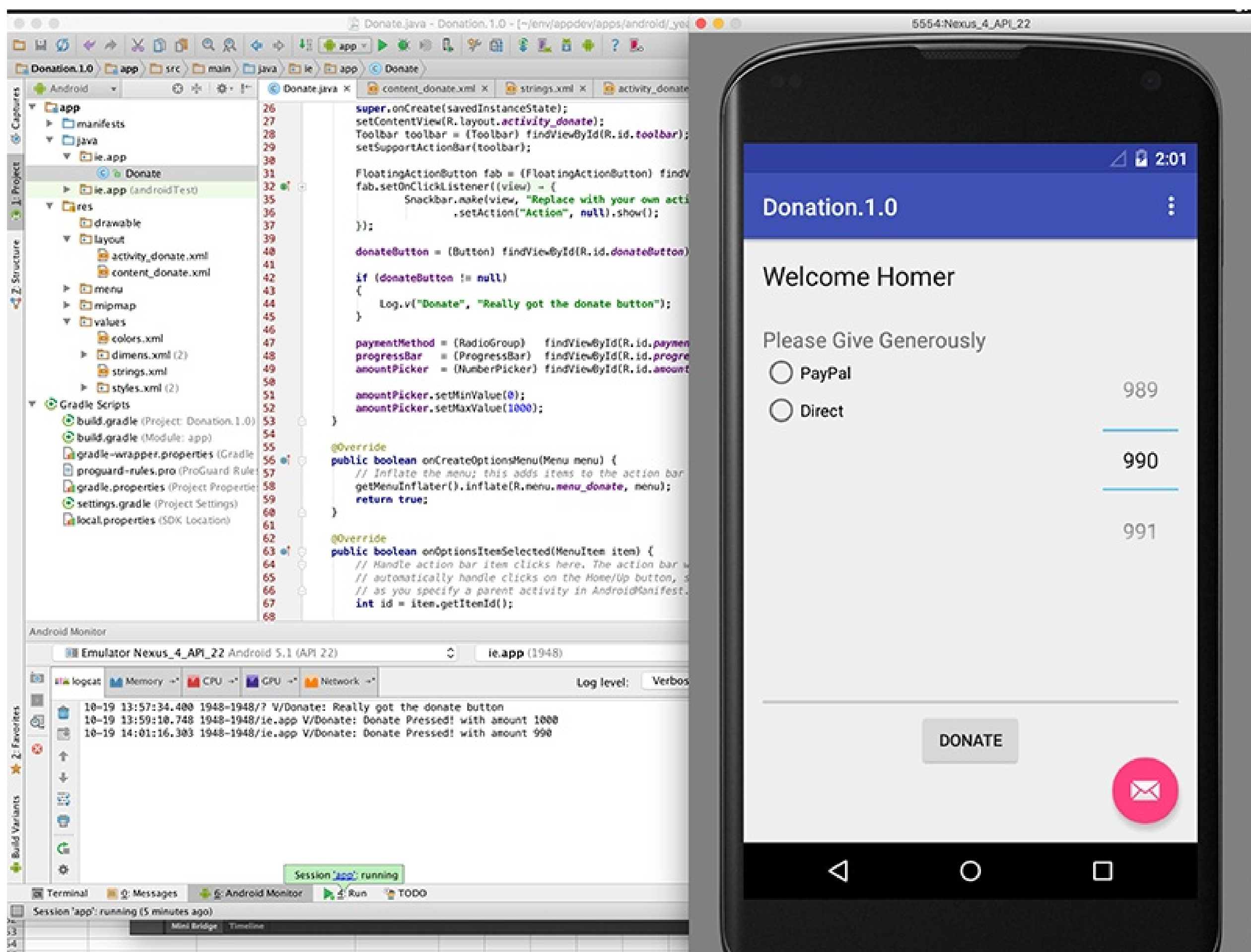
In onCreate, initialise the values:

```
    amountPicker.setMinValue(0);
    amountPicker.setMaxValue(1000);
```

And in donateButtonPressed:

```
public void donateButtonPressed (View view)
{
  int amount = amountPicker.getValue();
  Log.v("Donate", "Donate Pressed! with amount " + amount);
 }
```

Run this now - and verify that it operates as expected (see the actual amounts in the log file, as below).



Step 07 - The Radio Buttons

In *donateButtonPressed()* we need to discover which payment method has been selected. Our RadioGroup documentation is here:

- http://developer.android.com/reference/android/widget/RadioGroup.html

This looks like the method we need:

- getCheckedRadioButtonId())

This is a revised version of *donateButtonPressed()*

```
  public void donateButtonPressed (View view)
  {
    int amount = amountPicker.getValue();
    int radioId = paymentMethod.getCheckedRadioButtonId();
    String method = "";
    if (radioId == R.id.PayPal)
    {
      method = "PayPal";
    }
    else
    {
      method = "Direct";
    }
    Log.v("Donate", "Donate Pressed! with amount " + amount + ",
 method: " + method);
   }
```

Run it now and verify we are getting the correct logs.

We can simplify it somewhat by reducing the if statement to a single line:

```
    String method = radioId == R.id.PayPal ? "PayPal" : "Direct"
;
```

This is the Java ternary operator:

- http://marxsoftware.blogspot.ie/2010/09/how-i-learned-to-stop-worrying-and-love.html

This is the complete activity class so far:

```
package ie.app;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.NumberPicker;
import android.widget.ProgressBar;
import android.widget.RadioGroup;

public class Donate extends AppCompatActivity {

    private Button          donateButton;
    private RadioGroup      paymentMethod;
    private ProgressBar     progressBar;
    private NumberPicker    amountPicker;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_donate);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findVi
ewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
```

```
                Snackbar.make(view, "Replace with your own actio
n", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });

        donateButton = (Button) findViewById(R.id.donateButton);

        if (donateButton != null)
        {
            Log.v("Donate", "Really got the donate button");
        }

        paymentMethod = (RadioGroup)   findViewById(R.id.payment
Method);
        progressBar   = (ProgressBar)  findViewById(R.id.progres
sBar);
        amountPicker  = (NumberPicker) findViewById(R.id.amountP
icker);

        amountPicker.setMinValue(0);
        amountPicker.setMaxValue(1000);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar i
f it is present.
        getMenuInflater().inflate(R.menu.menu_donate, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar wi
ll
        // automatically handle clicks on the Home/Up button, so
 long
        // as you specify a parent activity in AndroidManifest.x
ml.
```

```
        int id = item.getItemId();

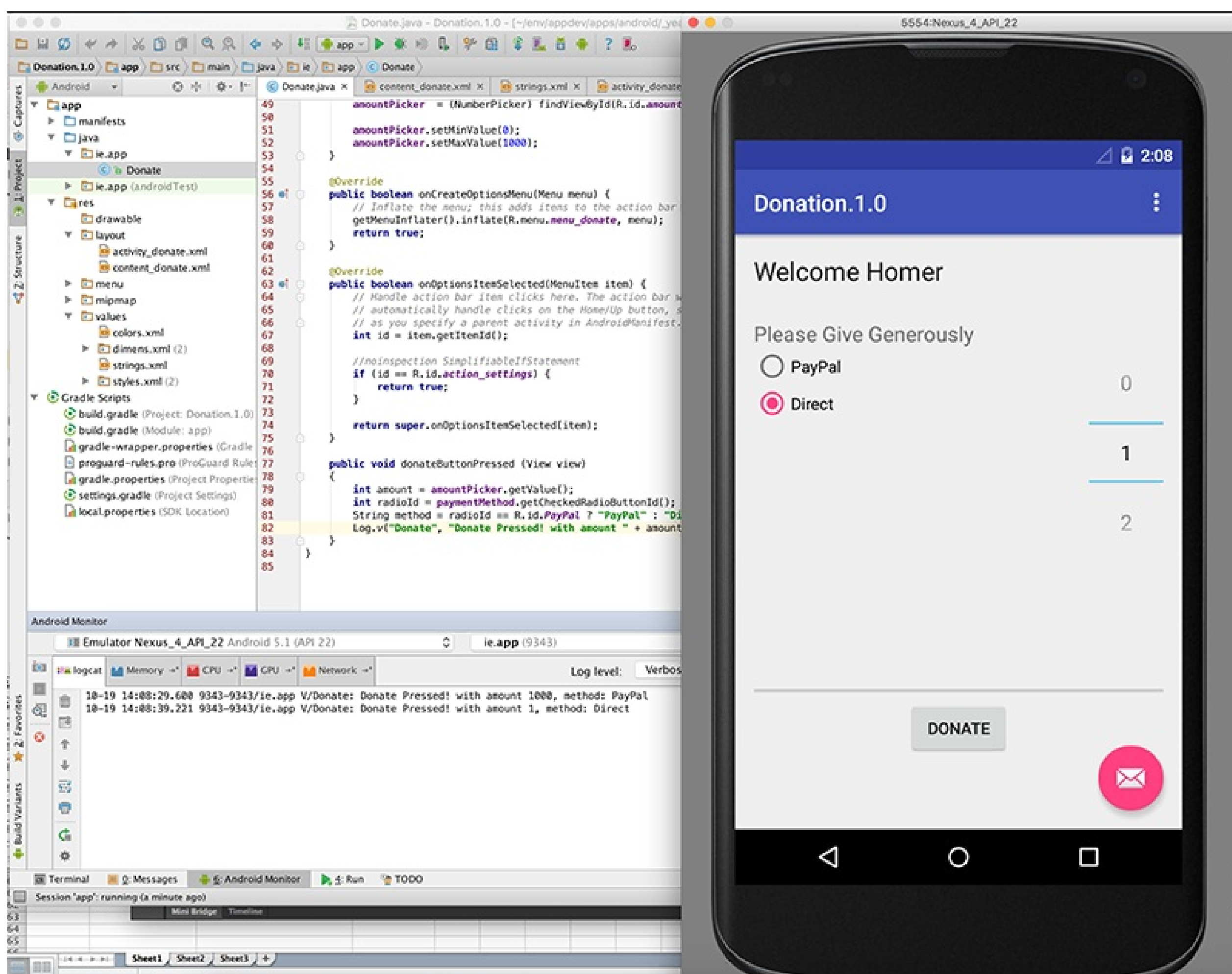
        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void donateButtonPressed (View view)
    {
        int amount = amountPicker.getValue();
        int radioId = paymentMethod.getCheckedRadioButtonId();
        String method = radioId == R.id.PayPal ? "PayPal" : "Dir
ect";
        Log.v("Donate", "Donate Pressed! with amount " + amount
+ ", method: " + method);
    }
}
```

So run your app again just to confirm the logCat entries

Step 07



5554:Nexus_4_API_22
2:08
Donation.1.0
Welcome Homer
Please Give Generously
PayPal
Direct
0
1
2
DONATE

Step 08 - The Progress Bar

The progress bar documentation:

- http://developer.android.com/reference/android/widget/ProgressBar.html

offers us advice on using the progress bar in multi-threaded application. Not quite what we are ready for yet! (but file it away for future reference).

These two methods are probably what we need:

- http://developer.android.com/reference/android/widget/ProgressBar.html#setMax(int)
- http://developer.android.com/reference/android/widget/ProgressBar.html#setProgress(int)

First we would need to equip our activity with the ability to remember the donation amounts:

```
  private int totalDonated = 0;
```

Lets set max progress bar to 10000 in onCreate:

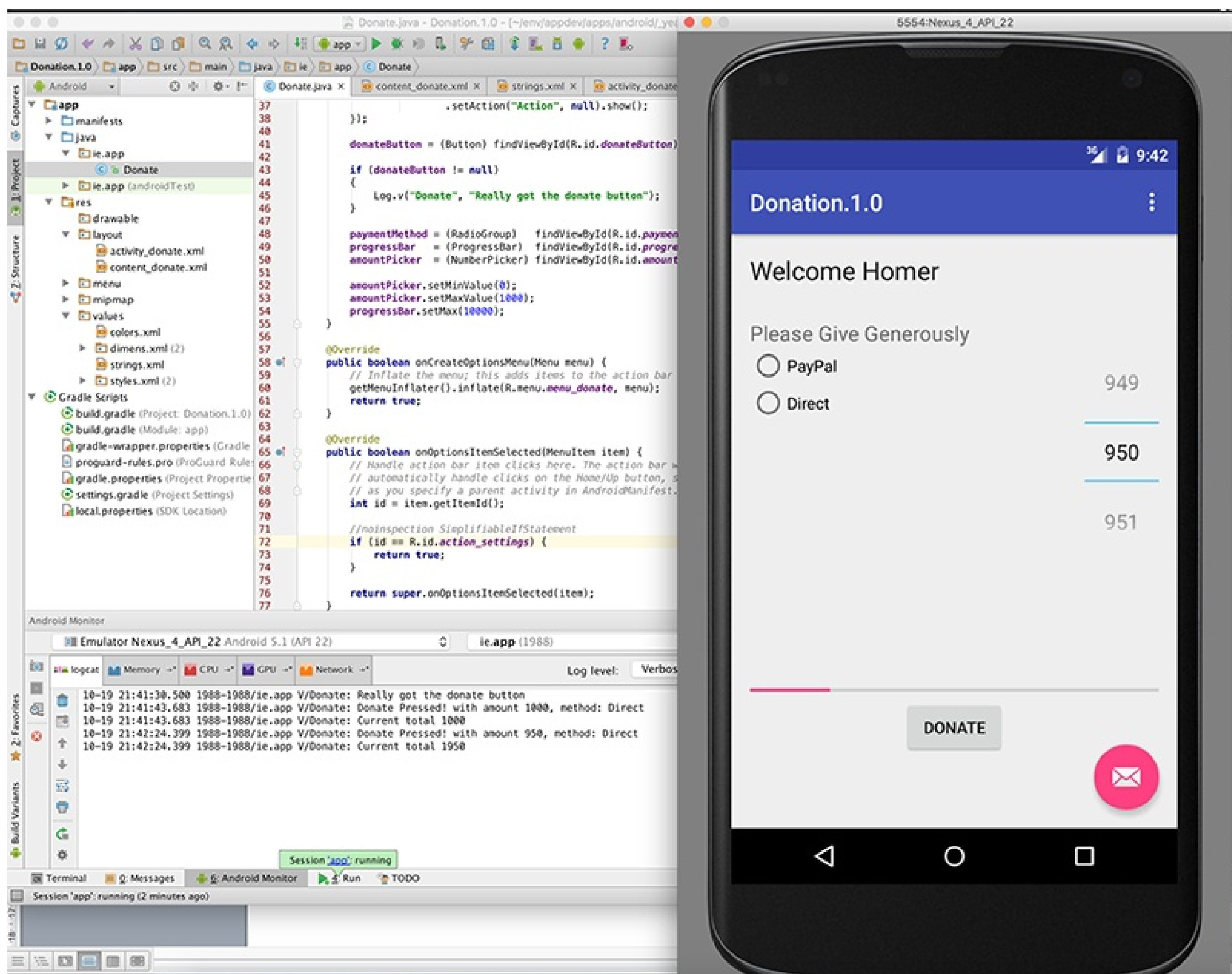
```
    progressBar.setMax(10000);
```

.. and set the progress in donateButtonPressed:

```
    totalDonated = totalDonated + amount;
    progressBar.setProgress(totalDonated);

    Log.v("Donate", "Donate Pressed! with amount " + amount + ", 
method: " + method);
    Log.v("Donate", "Current total " + totalDonated);
```

Try this now and observe the progress bar and logCat



This is the complete class so far:

```
package ie.app;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.NumberPicker;
import android.widget.ProgressBar;
import android.widget.RadioGroup;

public class Donate extends AppCompatActivity {
```

```
    private Button          donateButton;
    private RadioGroup      paymentMethod;
    private ProgressBar     progressBar;
    private NumberPicker    amountPicker;

    private int             totalDonated = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_donate);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findVi
ewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own actio
n", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });

        donateButton = (Button) findViewById(R.id.donateButton);

        if (donateButton != null)
        {
            Log.v("Donate", "Really got the donate button");
        }

        paymentMethod = (RadioGroup)   findViewById(R.id.payment
Method);
        progressBar   = (ProgressBar)  findViewById(R.id.progres
sBar);
        amountPicker  = (NumberPicker) findViewById(R.id.amountP
icker);
```

```
        amountPicker.setMinValue(0);
        amountPicker.setMaxValue(1000);
        progressBar.setMax(10000);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar i
f it is present.
        getMenuInflater().inflate(R.menu.menu_donate, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar wi
ll
        // automatically handle clicks on the Home/Up button, so
 long
        // as you specify a parent activity in AndroidManifest.x
ml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void donateButtonPressed (View view)
    {
        int amount = amountPicker.getValue();
        int radioId = paymentMethod.getCheckedRadioButtonId();
        String method = radioId == R.id.PayPal ? "PayPal" : "Dir
ect";

        totalDonated = totalDonated + amount;
        progressBar.setProgress(totalDonated);
```

```
        Log.v("Donate", "Donate Pressed! with amount " + amount
+ ", method: " + method);
        Log.v("Donate", "Current total " + totalDonated);
    }
}
```

Here is another version of exactly the same class:

```
package ie.app;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.NumberPicker;
import android.widget.ProgressBar;
import android.widget.RadioGroup;

public class Donate extends AppCompatActivity {

    private Button          donateButton;
    private RadioGroup      paymentMethod;
    private ProgressBar     progressBar;
    private NumberPicker    amountPicker;

    private int             totalDonated = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_donate);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
```

```
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findVi
ewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own actio
n", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });

        donateButton = (Button) findViewById(R.id.donateButton);

        if (donateButton != null)
        {
            Log.v("Donate", "Really got the donate button");
        }

        paymentMethod = (RadioGroup)   findViewById(R.id.payment
Method);
        progressBar   = (ProgressBar)  findViewById(R.id.progres
sBar);
        amountPicker  = (NumberPicker) findViewById(R.id.amountP
icker);

        amountPicker.setMinValue(0);
        amountPicker.setMaxValue(1000);
        progressBar.setMax(10000);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar i
f it is present.
        getMenuInflater().inflate(R.menu.menu_donate, menu);
        return true;
    }
```

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar wi
ll
        // automatically handle clicks on the Home/Up button, so
 long
        // as you specify a parent activity in AndroidManifest.x
ml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void donateButtonPressed (View view)
    {
    totalDonated  = totalDonated + amountPicker.getValue();
    String method = paymentMethod.getCheckedRadioButtonId() == R
.id.PayPal ? "PayPal" : "Direct";
    progressBar.setProgress(totalDonated);

    Log.v("Donate", amountPicker.getValue() + " donated by " +
method + "\nCurrent total " + totalDonated);
    }
}
```

Examine them carefully. What are the differences? Why make these changes?

Not also the careful attention to spacing and alignment in the code. Not just correct indentation, but continual attention to structuring each method carefully, removing duplication and unnecessary code and formatting/aligning the declarations and assignment statements in a table like structure:

Visible here:

```
private RadioGroup   paymentMethod;
private ProgressBar  progressBar;
private NumberPicker amountPicker;

private int          totalDonated = 0;
```

and here:

```
    paymentMethod = (RadioGroup)   findViewById(R.id.paymentMeth
od);
    progressBar   = (ProgressBar)  findViewById(R.id.progressBar
);
    amountPicker  = (NumberPicker) findViewById(R.id.amountPicke
r);
```

and here:

```
    totalDonated  = totalDonated + amountPicker.getValue();
    String method = paymentMethod.getCheckedRadioButtonId() == R
.id.PayPal ? "PayPal" : "Direct";
```

Android code can become very verbose and complex. Carefully formatting is essential if you are not to be overwhelmed.

Exercises

Archive of lab so far:

- Donation.1.0.zip

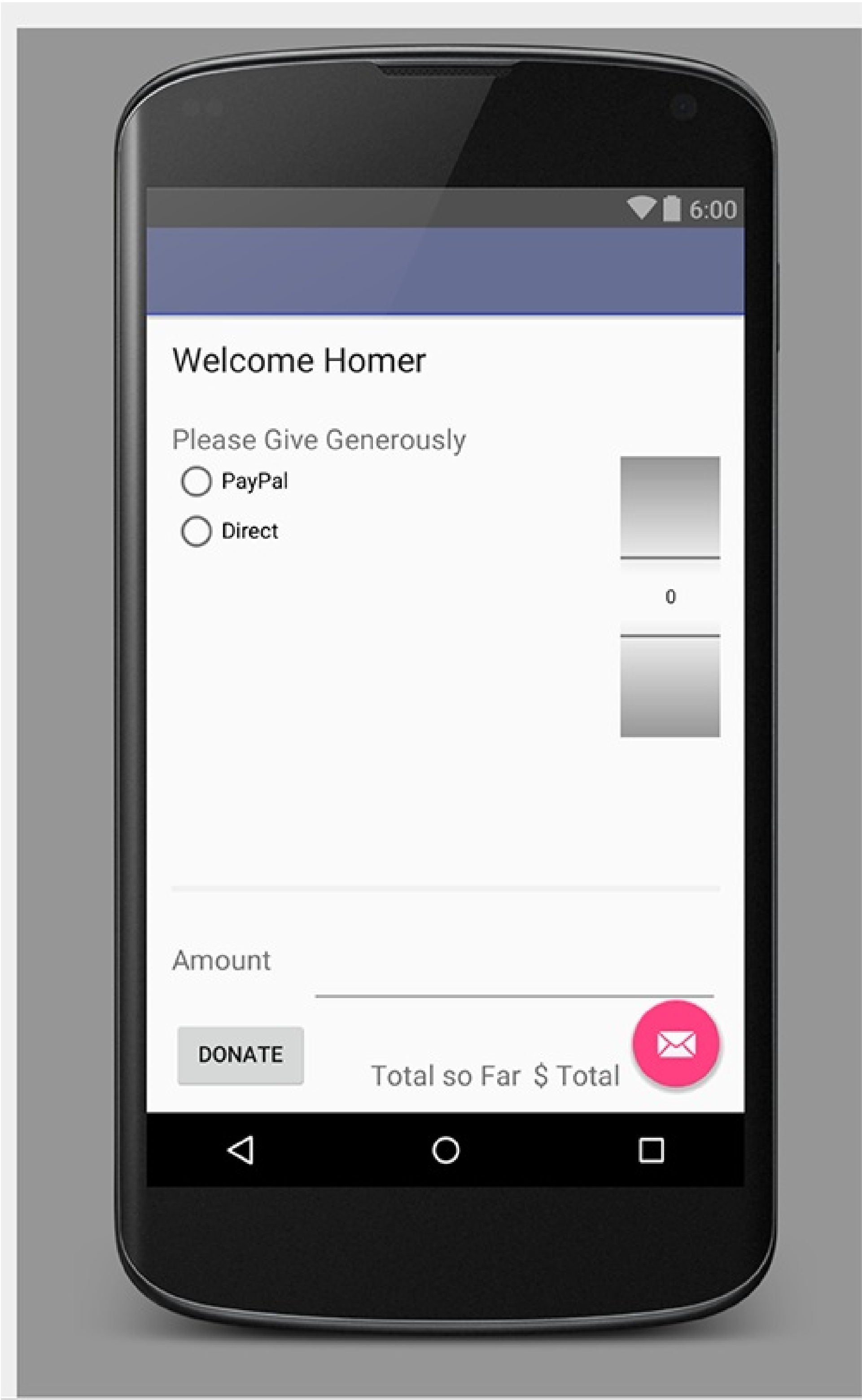
Exercise 1:

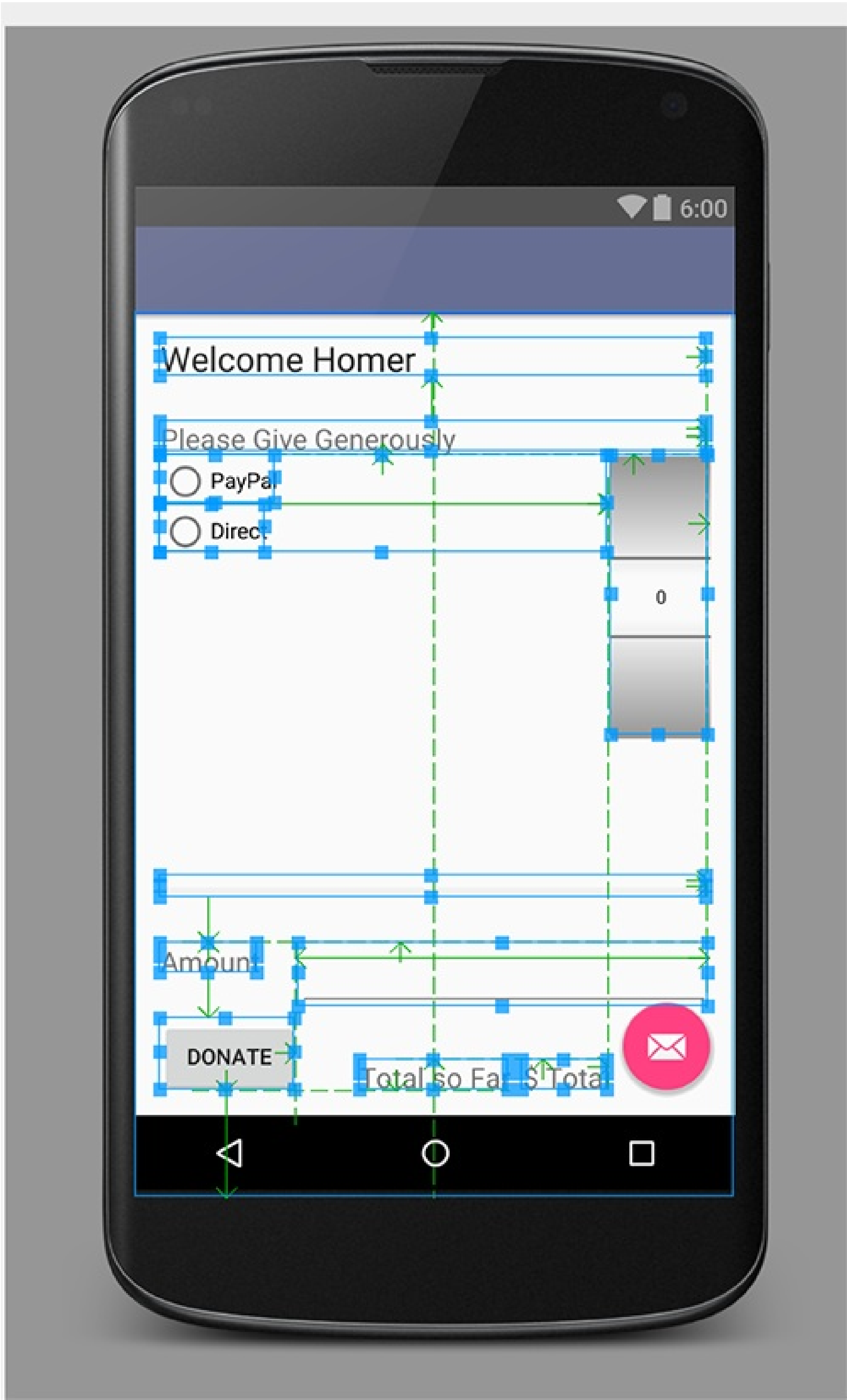
Consider an alternative to the NumberPicker - specifically one of the "Text Fields" controls:

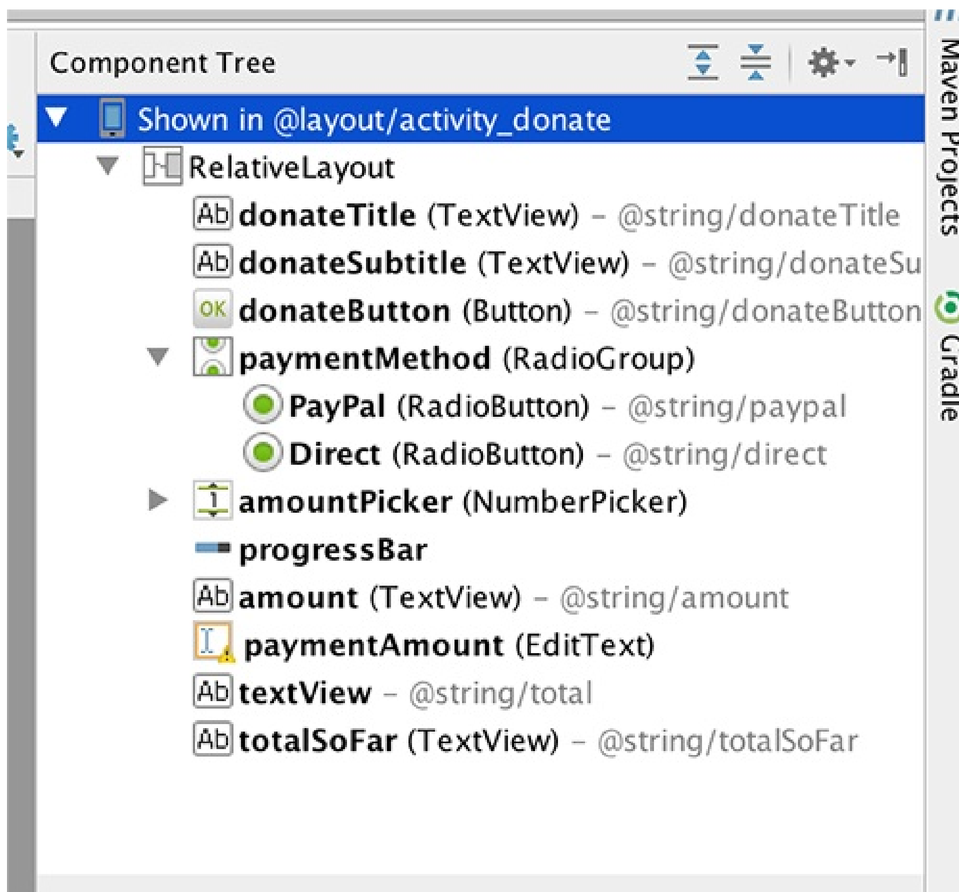
These are mostly EditView objects:

- http://developer.android.com/reference/android/widget/EditText.html

Redesign the activity to take a value from the picker or directly from a text view and maintain a "Total so Far" Value:


6:00
Welcome Homer
Please Give Generously
PayPal
Direct
0
Amount
DONATE
Total so Far $ Total


6:00
Welcome Homer
Please Give Generously
PayPal
Direct
0
Amount
DONATE
Total so Far $Total



If the number picker is set to zero, then attempt to get a number from the text view.

Here is a hint (a version of donatButonPressed that does what we want):

```
  public void donateButtonPressed (View view)
  {
    String method = paymentMethod.getCheckedRadioButtonId() == R
.id.PayPal ? "PayPal" : "Direct";
    progressBar.setProgress(totalDonated);

    int donatedAmount =  amountPicker.getValue();
    if (donatedAmount == 0)
    {
      String text = amountText.getText().toString();
      if (!text.equals(""))
        donatedAmount = Integer.parseInt(text);
    }
    totalDonated  = totalDonated + donatedAmount;
    Log.v("Donate", amountPicker.getValue() + " donated by " +
method + "\nCurrent total " + totalDonated);
   }
```

Exercise 2:

Revise the app such that when the target is achieved (10000) - then no more donations accepted, and the user is made aware of this.

Hint - here is how you can display a simple alert:

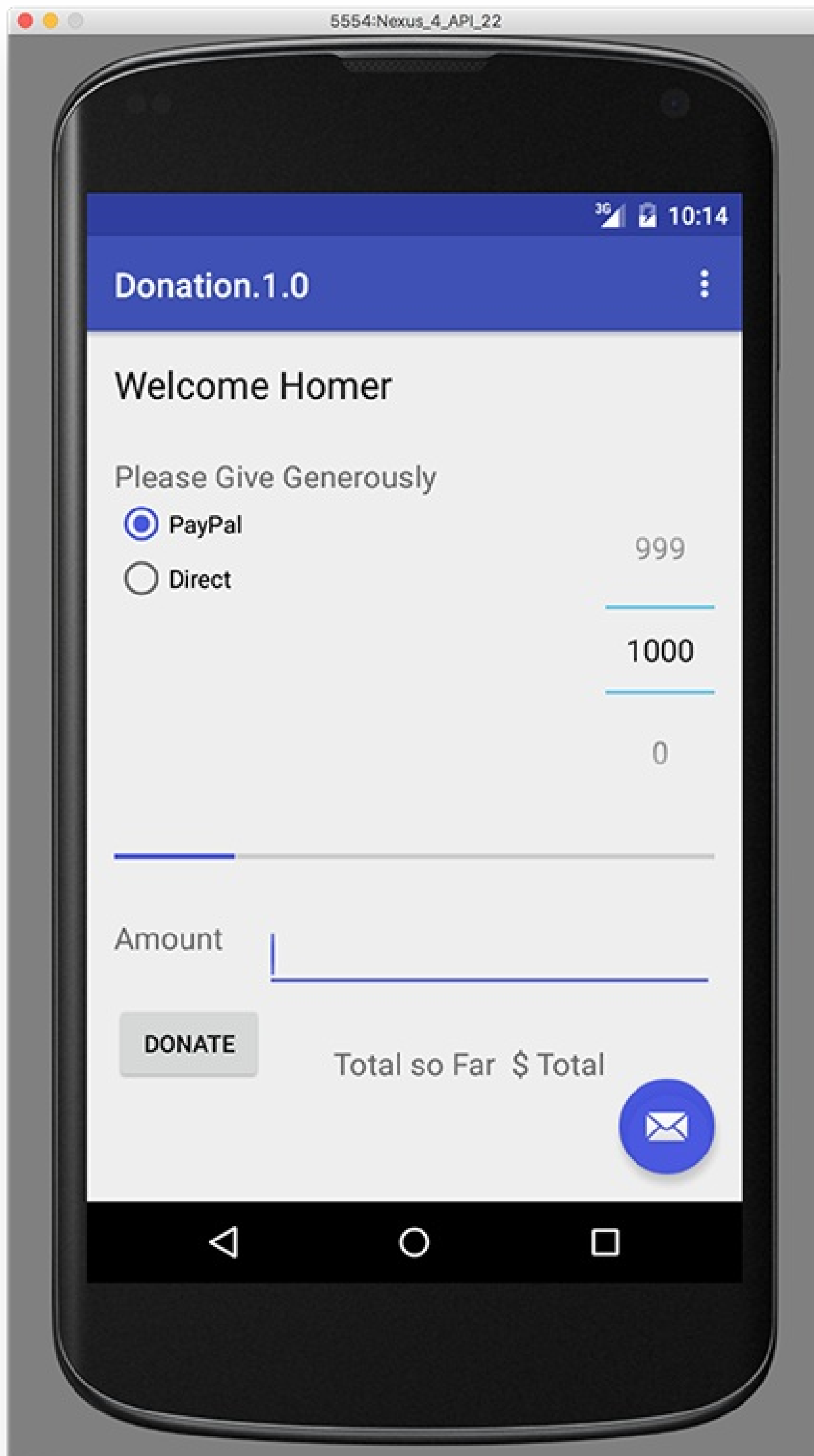
```
      Toast toast = Toast.makeText(this, "Target Exceeded!", Toa
st.LENGTH_SHORT);
      toast.show();
```

Exercise 3:

Modify the colour scheme for our widgets..

You will notice that the Floating Action Button, the Radio Buttons, the Progress Bar etc, are all a kind of pink - not really in line with our current colour scheme.

Hint - have a look at your **colors.xml**



Archive of lab with the above Exercises:

- Donation.1.5.zip

Lab 03: Donation 2.0 - Multi Screen App & Menus

Objectives

- Introduce simple menu handling
- Evolve the Donation android app to include a report view
- Use a simple ArrayAdapter

Package Name

This is the current version of the **Donation** app:

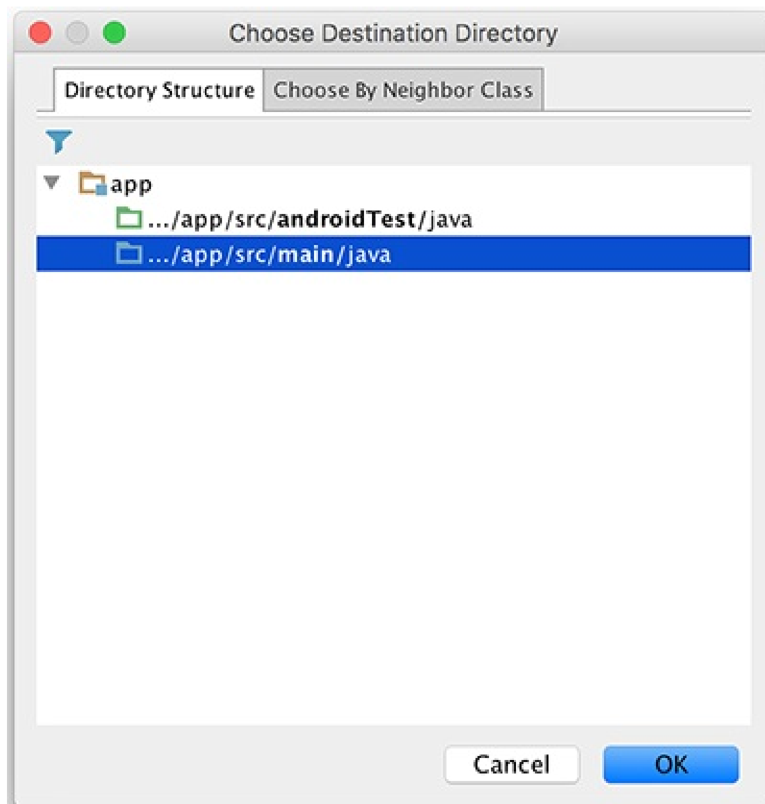
- Donation.1.5.zip

To continue using this project we need to 'refactor' it to **Donation.2.0**. At the time of writing, Android Studio's refactoring features and tools are a bit iffy :-) so we'll need to manually copy our project.

1. Ensure you don't have **Donation.1.5** open in Android Studio - if you do, close it now
2. Navigate to the folder where you downloaded and unzipped **Donation.1.5**
3. Rename (or copy if you wish) the folder to **Donation.2.0**
4. Rename the file **Donation.1.5.iml** to ***Donation.2.0.iml***
5. Edit ***Donation.2.0.iml*** and change any references to **Donation.1.5** to **Donation.2.0**
6. Navigate to the **.idea** folder (it might be a hidden folder) open the **.name** file and rename the project name to **Donation.2.0**
7. Launch Android Studio and open up the **Donation.2.0** project

We should also take this opportunity to change the name of the 'ie.app' package to 'ie.app.activities', as we will be introducing other packages later on.

So create a new Package (in the 'main' folder)



and drag in your **Donate.java** into this new package. You can delete the empty package (if Android Studio hasn't done it already)

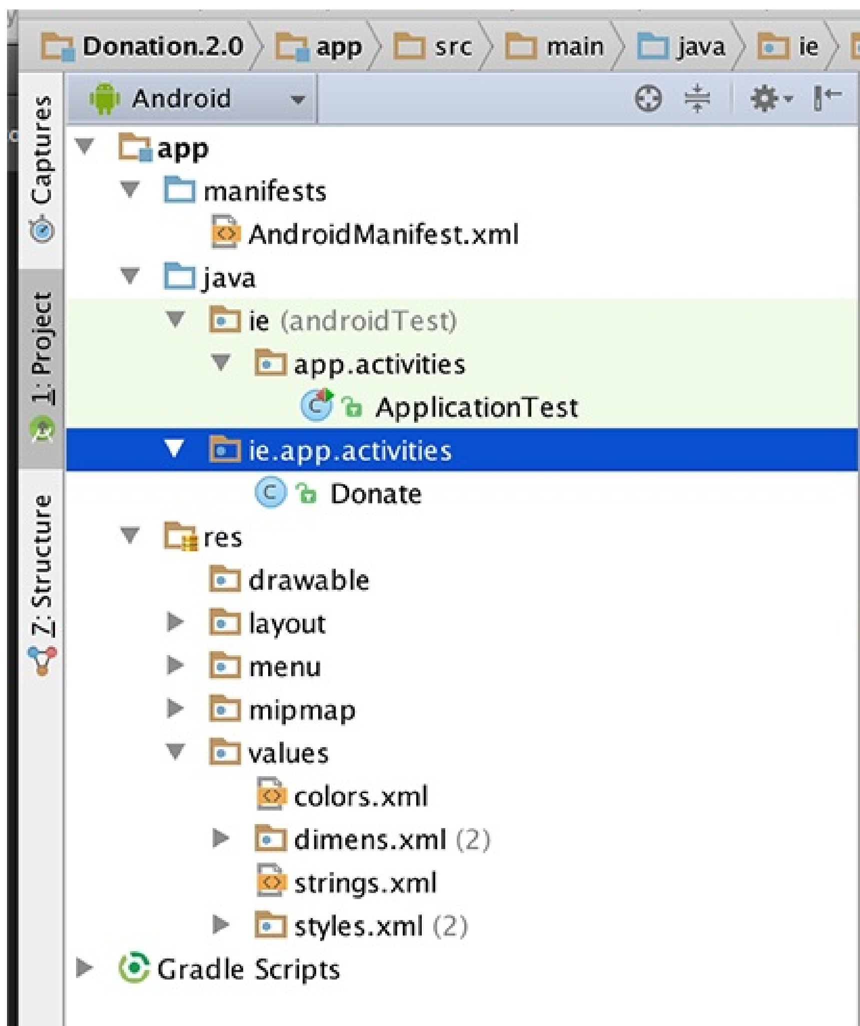
Do the same for the **ApplicationTest** class.

Next, check that the import statement in Donate.java is as follows:

```
import android.widget.Toast;

import ie.app.R;

public class Donate extends AppCompatActivity
```



So make sure your project structure looks like the above before continuing.

Also, don't forget to change the **app_name** string resource in your strings.xml.

You should 'Clean' your project at this stage "Build->Clean Project", and then Rebuild & finally run your app, to make sure everything is ok.

This is the current version of the **Donation** app (Your Starter Code for this lab, after you've completed the above):

- Donation.2.0.zip

Menu

Open **res/values/strings.xml** and introduce a new String resource like this:

```
<string name="menuReport">Report</string>
```

We have a menu resource called 'menu_donate.xml' in the res/menu folder. Modify this file by adding this new menu item:

```
  <item
       android:id="@+id/menuReport"
        android:orderInCategory="100"
        android:title="@string/menuReport"
        app:showAsAction="never"/>
```

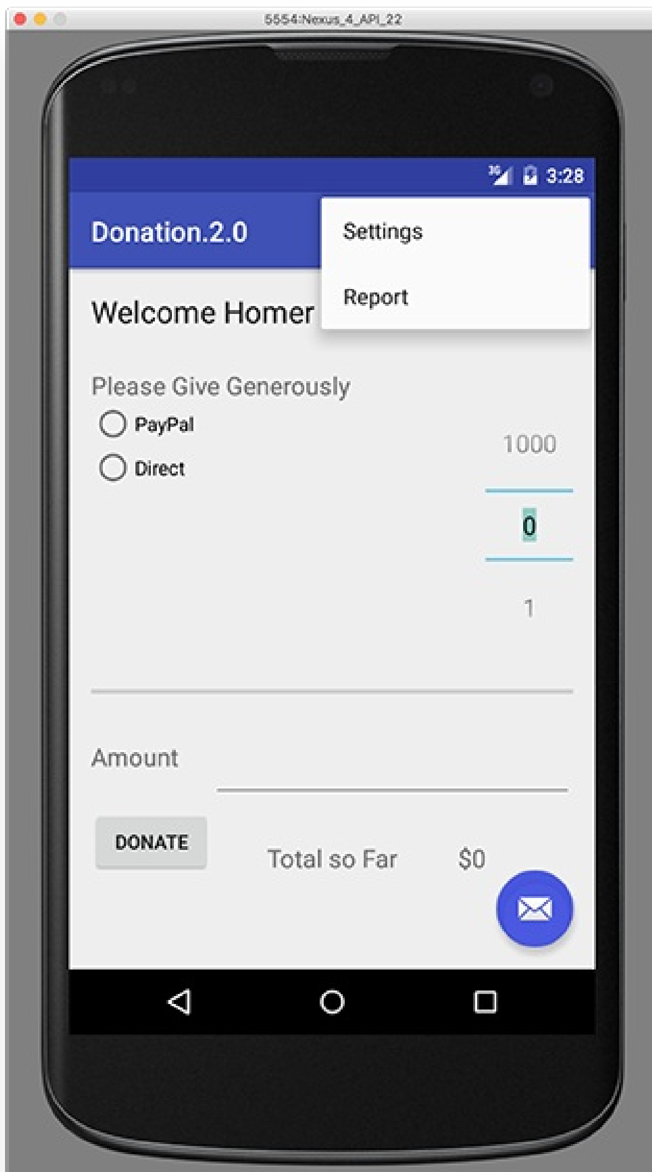
(Make sure it is within the **'menu'** element)

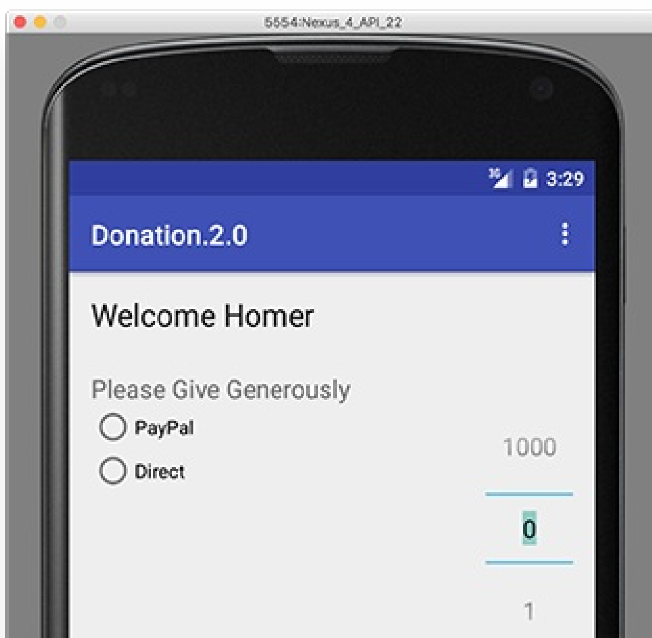
In Donate.java, change the onOptionsItemSelected method to look like this:

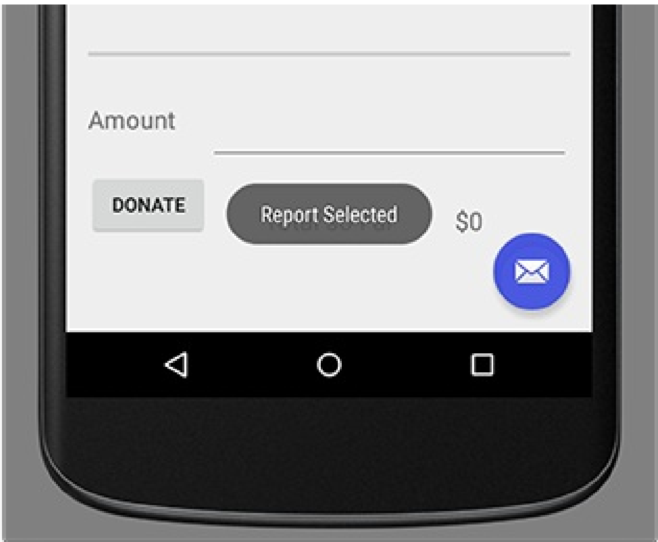
```
  @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId())
        {
            case R.id.menuReport:
                Toast toast = Toast.makeText(this, "Report Selec
ted", Toast.LENGTH_SHORT);
                toast.show();
                break;
        }

        return super.onOptionsItemSelected(item);
    }
```

Run the app and when you press the menu button (or the overflow menu) and select 'Report', you should see the toast message appear.


5554:Nexus_4_API_22
3:28
Donation.2.0
Settings
Report
Welcome Homer
Please Give Generously
PayPal
Direct
1000
0
1
Amount
DONATE
Total so Far
$0


5554:Nexus_4_API_22
3:29
Donation.2.0
Welcome Homer
Please Give Generously
PayPal
Direct
1000
0
1

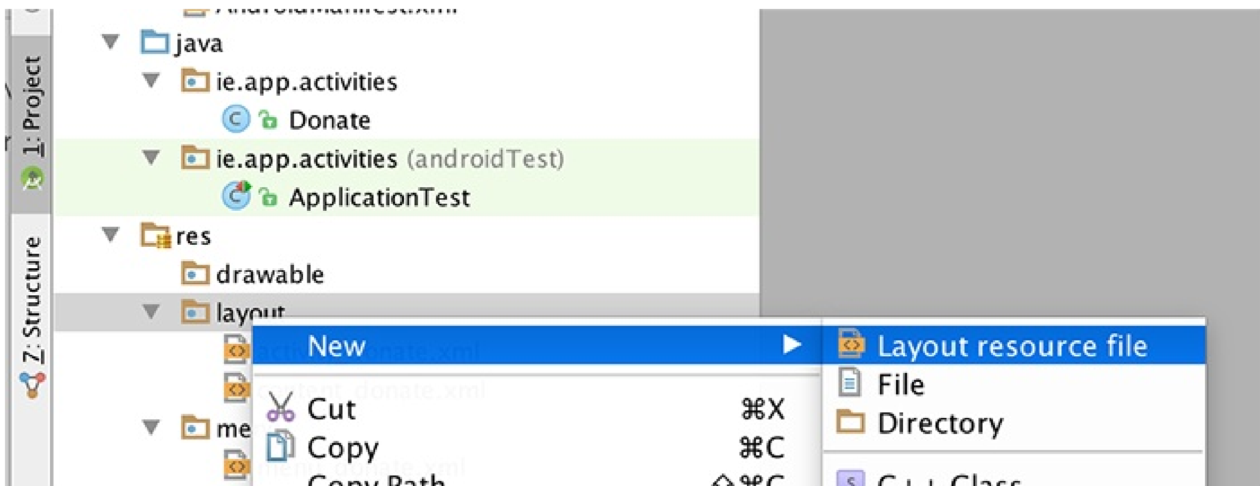

Amount
DONATE
Report Selected
$0

Reports Activity

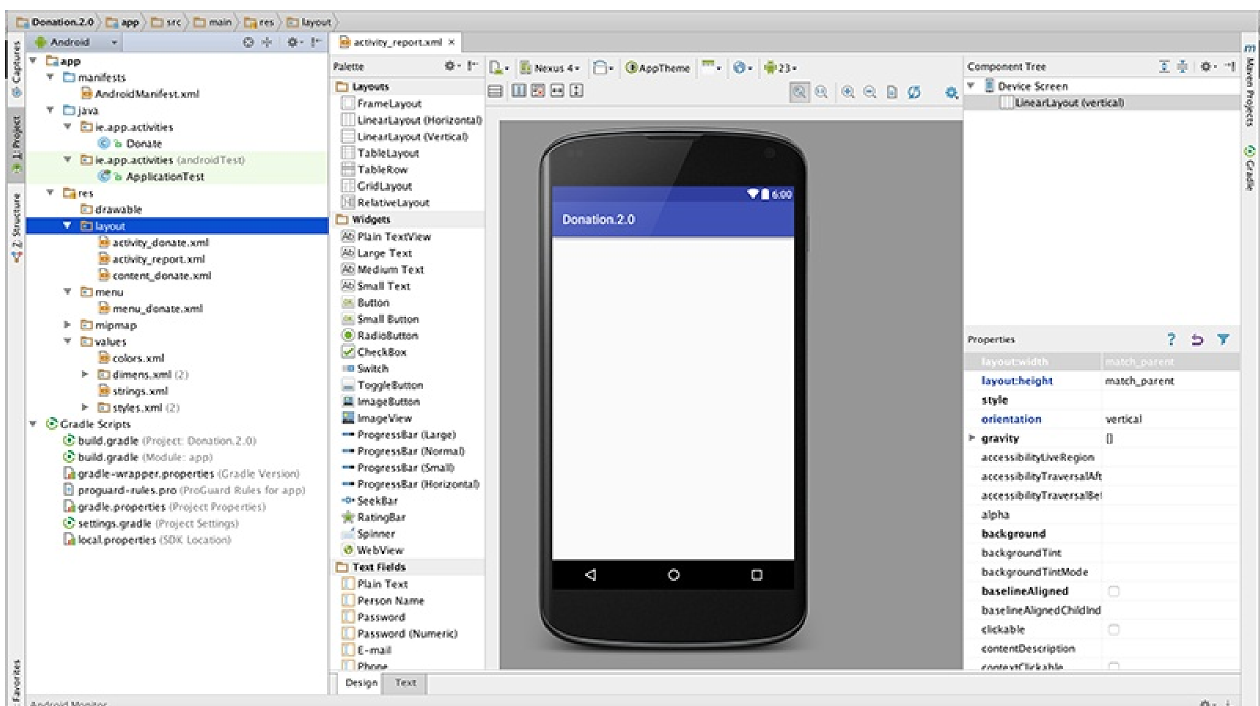
Before we start to design a new activity, Add a string resource in res/values/strings.xml:

```
<string name="reportTitle">Report</string>
```

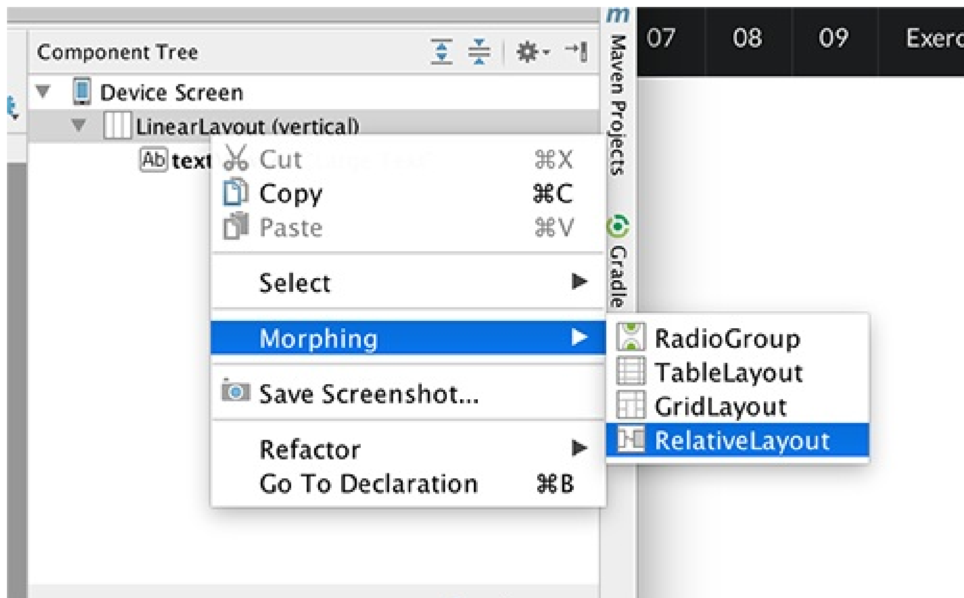
Design a new layout called **activity_report**. Do this by locating the res/layout folder and selecting new->layout resource file:



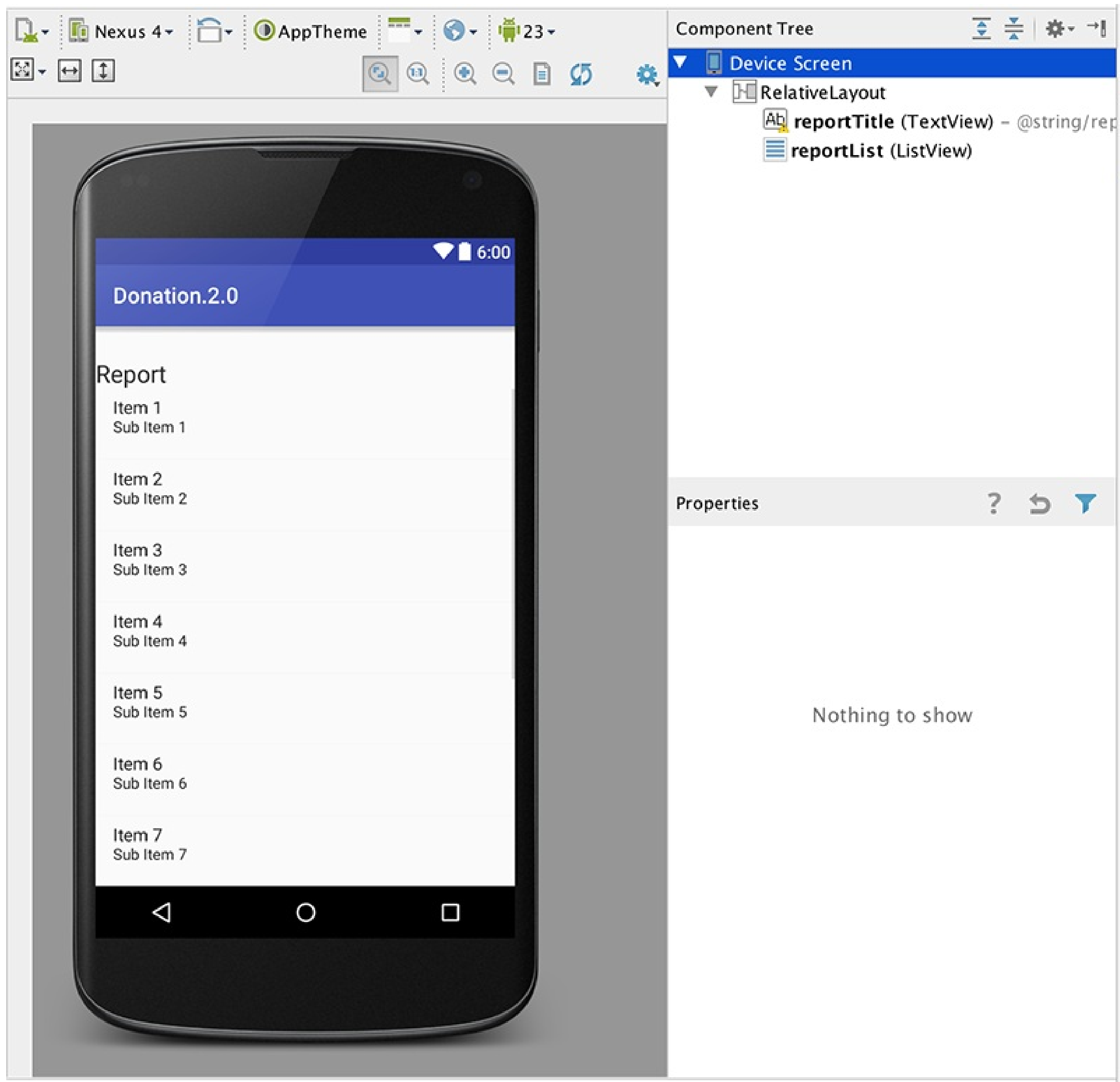
You can choose all the defaults for this layout.



'Morph' our layout from Linear to Relative, like so,



and build a layout similar to the following:



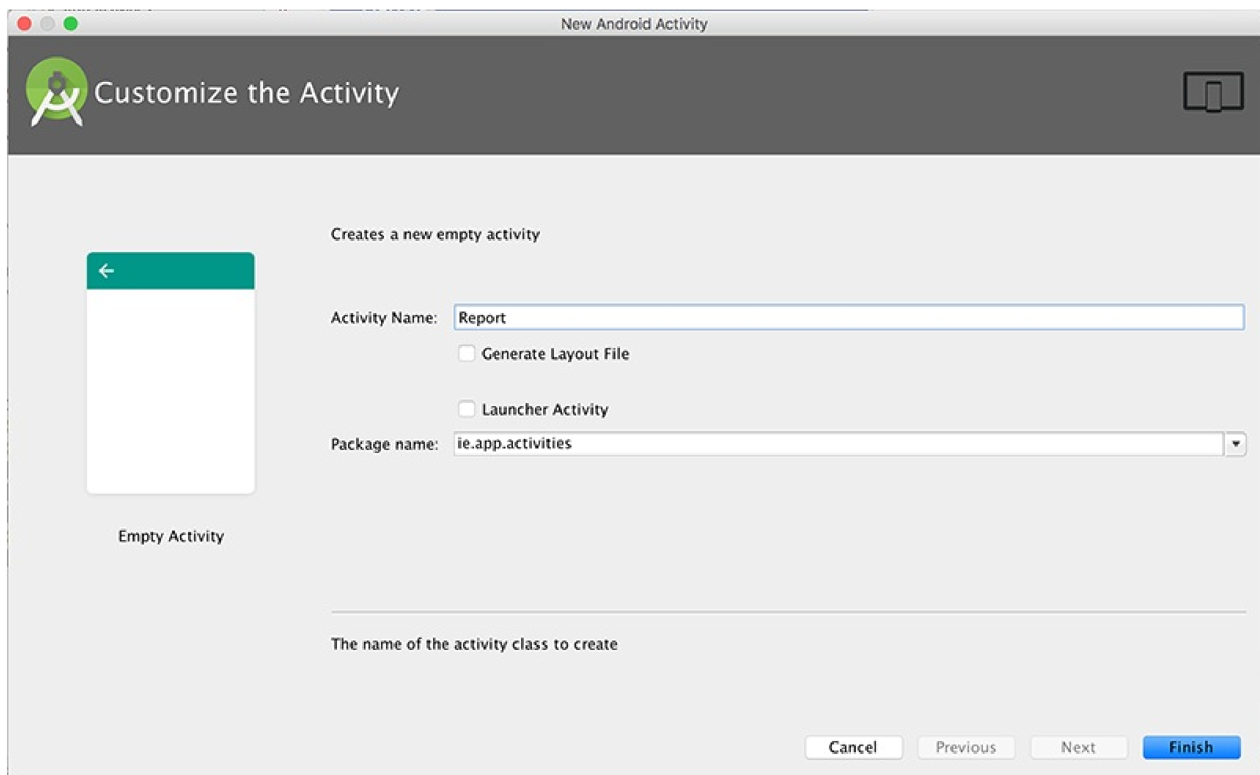
This is the layout file itself:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/re
s/android"
    android:orientation="vertical" android:layout_width="match_p
arent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarg
e"
        android:text="@string/reportTitle"
        android:id="@+id/reportTitle"
        android:layout_marginLeft="0dp"
        android:layout_marginTop="31dp"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true" />

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/reportList"
        android:layout_below="@+id/reportTitle"
        android:layout_alignParentStart="true" />
</RelativeLayout>
```

Introduce a new Empty Activity into ie.app.activities to render this activity:



and replace it with the following:

```
package ie.app.activities;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.app.Activity;
import android.widget.ArrayAdapter;
import android.widget.ListView;

import ie.app.R;

public class Report extends AppCompatActivity
{
  ListView listView;

  static final String[] numbers = new String[] {
      "Amount, Pay method",
      "10,     Direct",
      "100,    PayPal",
      "1000,   Direct",
      "10,     PayPal",
      "5000,   PayPal"};

  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_report);

    listView = (ListView) findViewById(R.id.reportList);
    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this
,  android.R.layout.simple_list_item_1, numbers);
    listView.setAdapter(adapter);
  }
}
```

This will display a hard-coded lists of donations.

Change Donation activity to load this view when 'Report' selected from menu:

```
  @Override
  public boolean onOptionsItemSelected(MenuItem item)
  {
    switch (item.getItemId())
    {
      case R.id.menuReport : startActivity (new Intent(this, Rep
ort.class));
                             break;
    }
   return super.onOptionsItemSelected(item);
  }
```

Confirm that the activity specification has been added to the AndroidManifest.xml file:

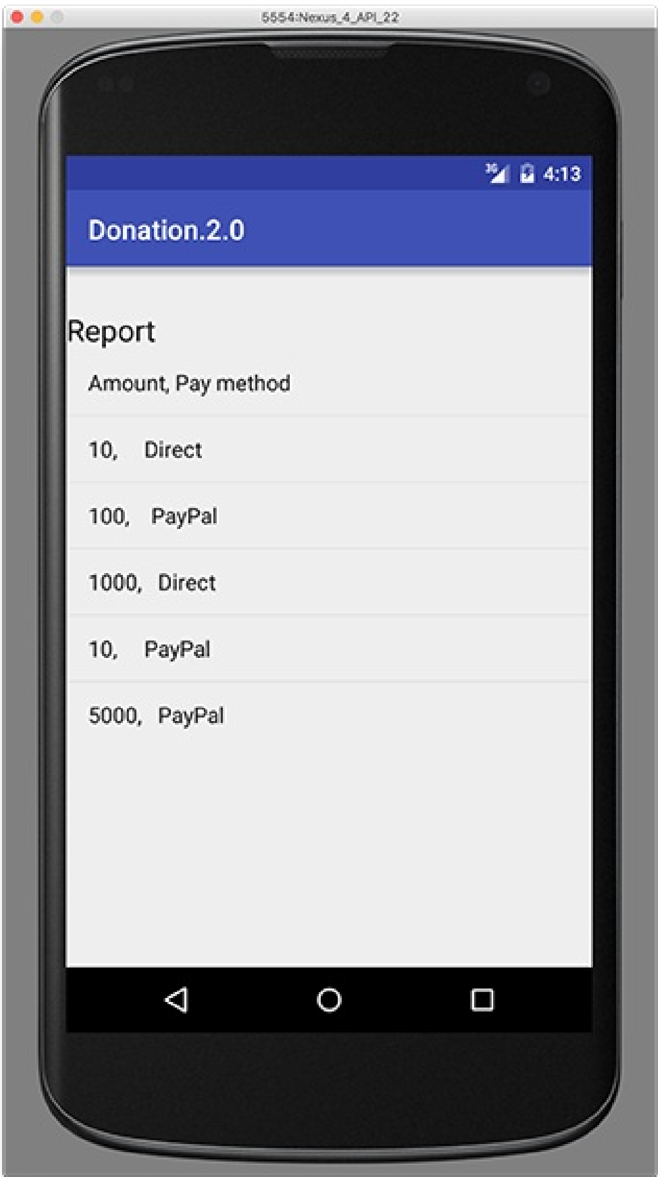
```
      <?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    package="ie.app" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".activities.Donate"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />

                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".activities.Report" >
        </activity>
    </application>

</manifest>
```

Try it all now - it should load (like below)


5554:Nexus_4_API_22
4:13
Donation.2.0
Report
Amount, Pay method
10, Direct
100, PayPal
1000, Direct
10, PayPal
5000, PayPal

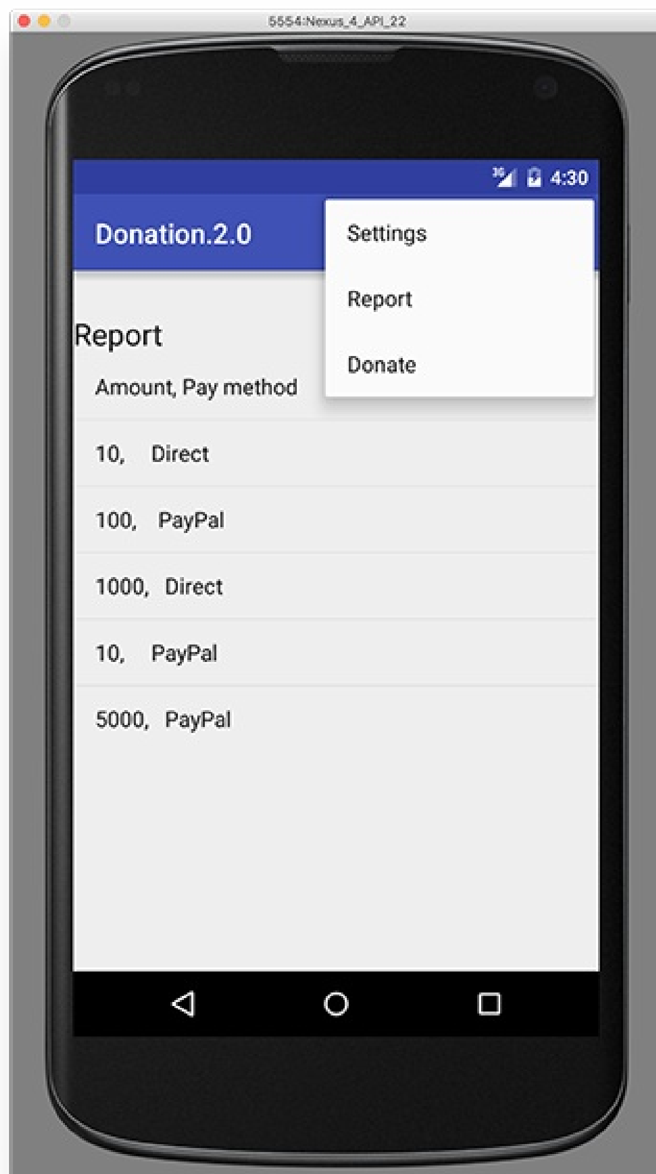
Menu Revisited

When you navigate from the Donate activity to reports, there will be no menu available. So we need to first inflate the menu like so

```
@Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar i
f it is present.
        getMenuInflater().inflate(R.menu.menu_donate, menu);
        return true;
    }
```

Bring in a new menu item, 'Donate' - Donate should bring you back to the main Donate Screen.

This is a very similar approach to what you did in Step 02, so revisit this step and see if you can end up with something like the following for the 'Report' Screen:



We'll look at populating the list with actual donations in the next lab.

Solution

Solution so far:

- Donation.2.0.zip

Lab 04: Donation 3.0 - Donation Object Model

Objectives

- Evolve the Donation android app to include a Donation Object Model and Base class
- Use a CustomAdapter in the Report View.
- Refactor our menus with xml driven event handling

Donation Model & Base Class

In order to keep our application design coherent, we now bring in an Base class and a Donation class to manage our Donations. You can continue with your own version of the app or start with the solution from the previous lab - Donation.2.0

But before we do anything, it's probably a good idea to rename/copy or project to **Donation.3.0** like we did in the previous lab - so go ahead and do that now (just make sure it's not open Android Studio!)

Now, first, create a new package called 'ie.app.models' in 'main' and bring in this class here:

```
package ie.app.models;

public class Donation
{
  public int    amount;
  public String method;

  public Donation (int amount, String method)
  {
    this.amount = amount;
    this.method = method;
  }
}
```

Next, Create a new class called 'Base' and add it to the 'ie.app.activities' package: (and fix the errors obviously :-))

```
public class Base extends AppCompatActivity
{
  public final int       target       = 10000;
  public int             totalDonated = 0;
  public static List <Donation> donations    = new ArrayList<Don
ation>();
```

```
  public boolean newDonation(Donation donation)
  {
    boolean targetAchieved = totalDonated > target;
    if (!targetAchieved)
    {
      donations.add(donation);
      totalDonated += donation.amount;
    }
    else
    {
      Toast toast = Toast.makeText(this, "Target Exceeded!", Toa
st.LENGTH_SHORT);
      toast.show();
    }
    return targetAchieved;
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu)
  {
    getMenuInflater().inflate(R.menu.donate, menu);
    return true;
  }

  @Override
  public boolean onPrepareOptionsMenu (Menu menu){
      super.onPrepareOptionsMenu(menu);
      MenuItem report = menu.findItem(R.id.menuReport);
      MenuItem donate = menu.findItem(R.id.menuDonate);

      if(donations.isEmpty())
           report.setEnabled(false);
      else
        report.setEnabled(true);

      if(this instanceof Donate){
          donate.setVisible(false);
          if(!donations.isEmpty())
                report.setVisible(true);
        }
```

```
      else {
        report.setVisible(false);
        donate.setVisible(true);
        }

      return true;
  }

  public void settings(MenuItem item)
  {
    Toast.makeText(this, "Settings Selected", Toast.LENGTH_SHORT
).show();
  }

  public void report(MenuItem item)
  {
    startActivity (new Intent(this, Report.class));
  }

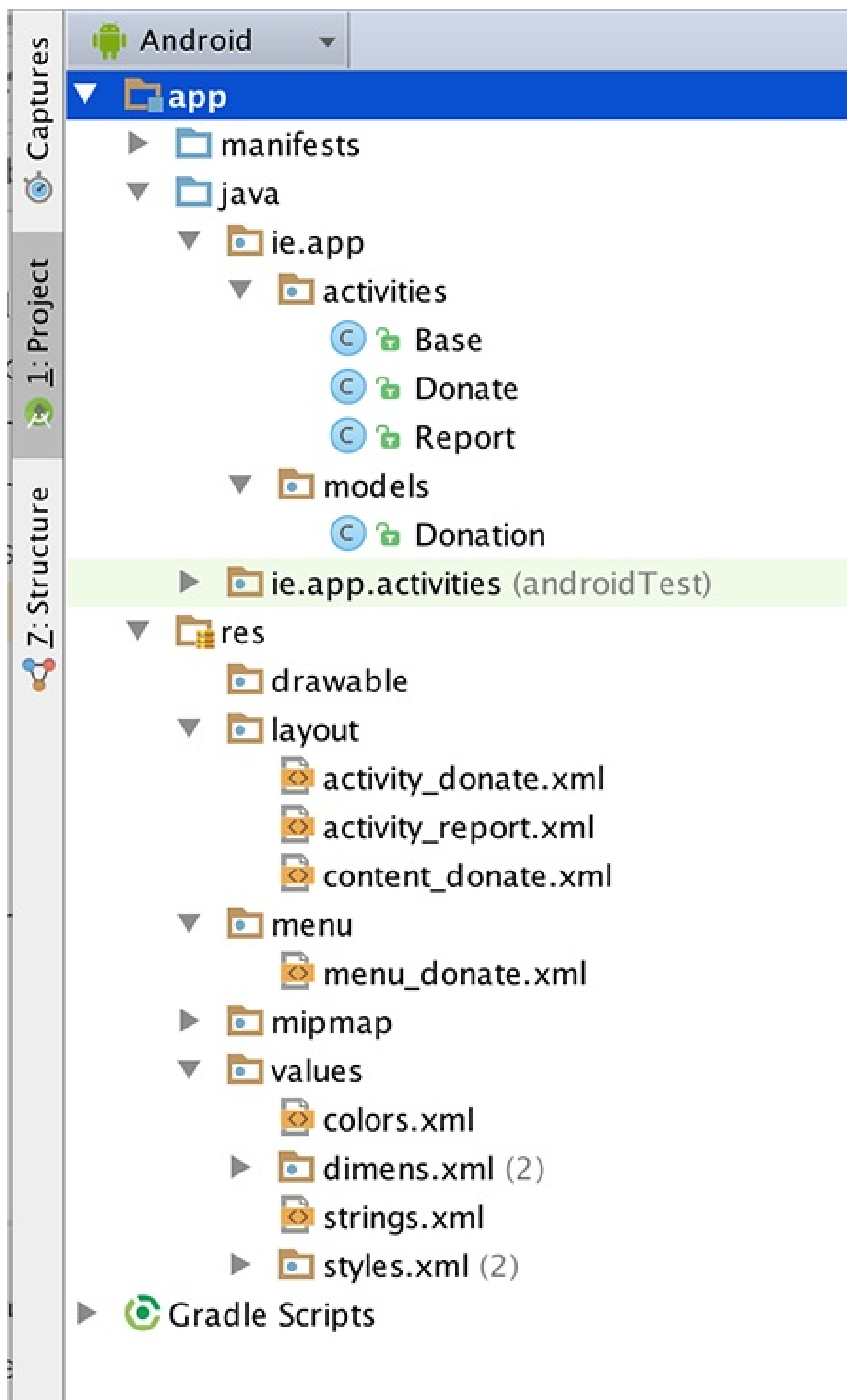
  public void donate(MenuItem item)
  {
    startActivity (new Intent(this, Donate.class));
  }
}
```

Notice our List of Donations in the Base class - we will use this list to display our Donations in the Report.

There is still one error remaining, can you work out why, and more importantly, how to fix it?

Hint - it relates to class inheritance, and some refactoring that needs to be done.

Your project should now look as follows:



Also, pay particular attention to the 'settings', 'report' and 'donate' methods - these will be triggered directly by our menu options via xml, which we'll look at in Step 05.

Refactored Donate

The Donate activity can now be completely refactored to make use of the Base class.

```
public class Donate extends Base {

    private Button          donateButton;
    private RadioGroup      paymentMethod;
    private ProgressBar     progressBar;
    private NumberPicker    amountPicker;
    private EditText        amountText;
    private TextView        amountTotal;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_donate);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findVi
ewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own actio
n", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });

        donateButton = (Button) findViewById(R.id.donateButton);

        paymentMethod = (RadioGroup)   findViewById(R.id.payment
Method);
        progressBar   = (ProgressBar)  findViewById(R.id.progres
sBar);
```

```
        amountPicker  = (NumberPicker) findViewById(R.id.amountP
icker);
        amountText    = (EditText)     findViewById(R.id.payment
Amount);
        amountTotal   = (TextView)     findViewById(R.id.totalSo
Far);

        amountPicker.setMinValue(0);
        amountPicker.setMaxValue(1000);
        progressBar.setMax(10000);
        amountTotal.setText("$0");
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar i
f it is present.
        getMenuInflater().inflate(R.menu.menu_donate, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
        switch (item.getItemId())
        {
            case R.id.menuReport : startActivity (new Intent(this
, Report.class));
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    public void donateButtonPressed (View view)
    {
        String method = paymentMethod.getCheckedRadioButtonId()
== R.id.PayPal ? "PayPal" : "Direct";
        int donatedAmount =  amountPicker.getValue();
        if (donatedAmount == 0)
        {
```

```
            String text = amountText.getText().toString();
            if (!text.equals(""))
                donatedAmount = Integer.parseInt(text);
        }
        if (donatedAmount > 0)
        {
            newDonation(new Donation(donatedAmount, method));
            progressBar.setProgress(totalDonated);
            String totalDonatedStr = "$" + totalDonated;
            amountTotal.setText(totalDonatedStr);
        }
    }
}
```

Replace your version with this and execute it - fix any missing import statements necessary.

Look carefully at the changes to this version over the previous attempt.

Refactored Report - New 'Layout'

We now rework Report to render the actual donations - held in the Base class list.

First some layout additions. Include these new string resources in strings.xml

```
    <string name="defaultAmount">00</string>
    <string name="defaultMethod">N/A</string>
```

This is a new layout - to be called 'row_donate.xml'. Place this in the 'layout' folder.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/re
s/android"
    android:orientation="vertical" android:layout_width="match_p
arent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/row_amount"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="48dp"
        android:layout_marginTop="20dp"
        android:text="@string/defaultAmount" />

    <TextView
        android:id="@+id/row_method"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="106dp"
        android:text="@string/defaultMethod"
        android:layout_alignTop="@+id/row_amount"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```

Refactored Report - New 'Class'

Finally, rework the Report class to remove the hard coded values - and use a different 'adapter'

```
public class Report extends Base
{
    ListView listView;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_report);

        listView = (ListView) findViewById(R.id.reportList);
        DonationAdapter adapter = new DonationAdapter(this,  don
ations);
        listView.setAdapter(adapter);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar i
f it is present.
        getMenuInflater().inflate(R.menu.menu_donate, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
        switch (item.getItemId())
        {
            case R.id.menuDonate : startActivity (new Intent(this
, Donate.class));
                break;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

This is the new adapter - DonationAdapter. You can place this at the end of the Report class (outside the closing brace) if you like:

```
class DonationAdapter extends ArrayAdapter<Donation>
{
  private Context        context;
  public  List<Donation> donations;

  public DonationAdapter(Context context, List<Donation> donations)
  {
    super(context, R.layout.row_donate, donations);
    this.context   = context;
    this.donations = donations;
  }

  @Override
  public View getView(int position, View convertView, ViewGroup parent)
  {
    LayoutInflater inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

    View     view       = inflater.inflate(R.layout.row_donate, parent, false);
    Donation donation   = donations.get(position);
    TextView amountView = (TextView) view.findViewById(R.id.row_amount);
    TextView methodView = (TextView) view.findViewById(R.id.row_method);

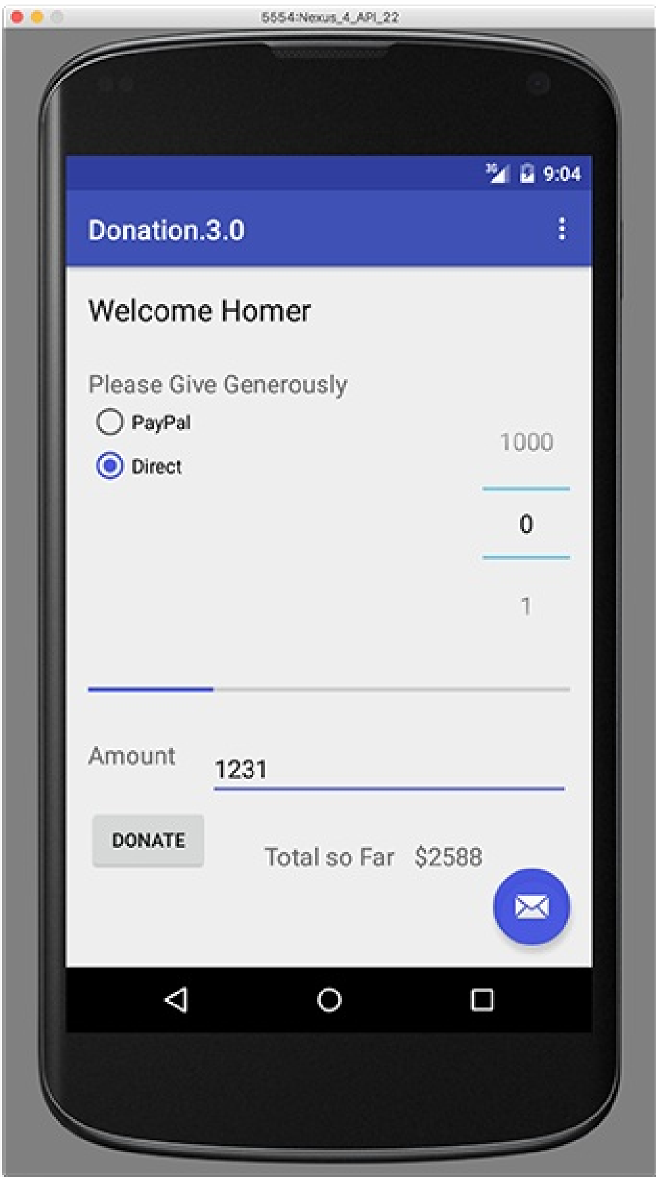
    amountView.setText("$" + donation.amount);
    methodView.setText(donation.method);

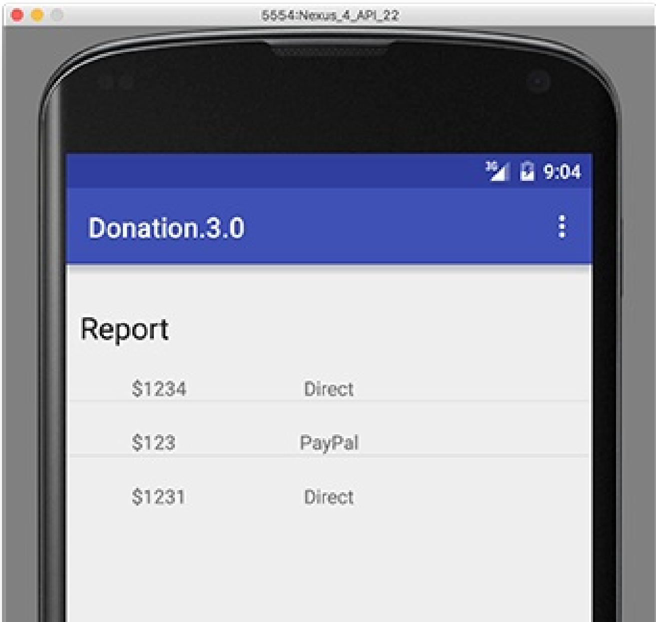
    return view;
  }

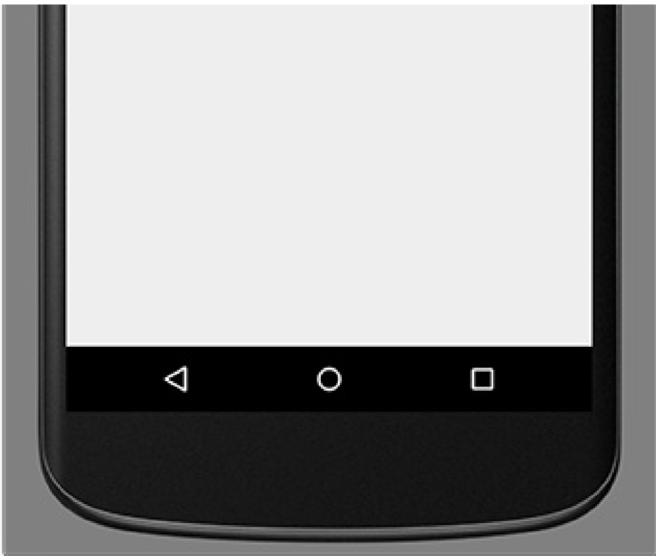
  @Override
  public int getCount()
  {
```

```
      return donations.size();
    }
  }
```

If all goes well - then you should be able to make donations, and then see a list of them in the report activity.


5554:Nexus_4_API_22
9:04
Donation.3.0
Welcome Homer
Please Give Generously
PayPal
Direct
1000
0
1
Amount
1231
DONATE
Total so Far $2588


5554:Nexus_4_API_22
9:04
Donation.3.0
Report
$1234 Direct
$123 PayPal
$1231 Direct



Refactored Menu - New 'Event Handling'

If you recall from Step 01, our Base class has 3 particular methods that we haven't made use of - yet

```
  public void settings(MenuItem item)
  {
    Toast.makeText(this, "Settings Selected", Toast.LENGTH_SHORT
).show();
  }

  public void report(MenuItem item)
  {
    startActivity (new Intent(this, Report.class));
  }

  public void donate(MenuItem item)
  {
    startActivity (new Intent(this, Donate.class));
  }
}
```

We will now use these methods, and using the **onClick** attribute in our MenuItems, we will 'bind' these methods to our menu options.

Firstly, edit your menu_donate.xml and ensure it now looks as follows:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools" tools:context=
".Donate">

    <item android:id="@+id/action_settings"
        android:title="@string/action_settings"
        android:orderInCategory="100"
        app:showAsAction="never"
        android:onClick="settings"/>

    <item
        android:id="@+id/menuReport"
        android:orderInCategory="100"
        android:title="@string/menuReport"
        app:showAsAction="never"
        android:onClick="report"/>

    <item
        android:id="@+id/menuDonate"
        android:orderInCategory="100"
        android:title="@string/menuDonate"
        app:showAsAction="never"
        android:onClick="donate"/>
</menu>
```

note the added onClick attributes for each MenuItem, it directly corresponds to the method names in our Base class

We can now refactor our Donate & Report classes and **Remove** the menu inflation and event handling code in both classes.

So, both classes should look as follows:

Donate

```
public class Donate extends Base {

    private Button          donateButton;
```

```
    private RadioGroup      paymentMethod;
    private ProgressBar     progressBar;
    private NumberPicker    amountPicker;
    private EditText        amountText;
    private TextView        amountTotal;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_donate);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findVi
ewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own actio
n", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });

        donateButton = (Button) findViewById(R.id.donateButton);

        paymentMethod = (RadioGroup)   findViewById(R.id.payment
Method);
        progressBar   = (ProgressBar)  findViewById(R.id.progres
sBar);
        amountPicker  = (NumberPicker) findViewById(R.id.amountP
icker);
        amountText    = (EditText)     findViewById(R.id.payment
Amount);
        amountTotal   = (TextView)     findViewById(R.id.totalSo
Far);

        amountPicker.setMinValue(0);
        amountPicker.setMaxValue(1000);
        progressBar.setMax(10000);
```

```
        amountTotal.setText("$0");
    }

    public void donateButtonPressed (View view)
    {
        String method = paymentMethod.getCheckedRadioButtonId()
== R.id.PayPal ? "PayPal" : "Direct";
        int donatedAmount =  amountPicker.getValue();
        if (donatedAmount == 0)
        {
            String text = amountText.getText().toString();
            if (!text.equals(""))
                donatedAmount = Integer.parseInt(text);
        }
        if (donatedAmount > 0)
        {
            newDonation(new Donation(donatedAmount, method));
            progressBar.setProgress(totalDonated);
            String totalDonatedStr = "$" + totalDonated;
            amountTotal.setText(totalDonatedStr);
        }
    }
}
```

Report

```
public class Report extends Base
{
    ListView listView;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_report);

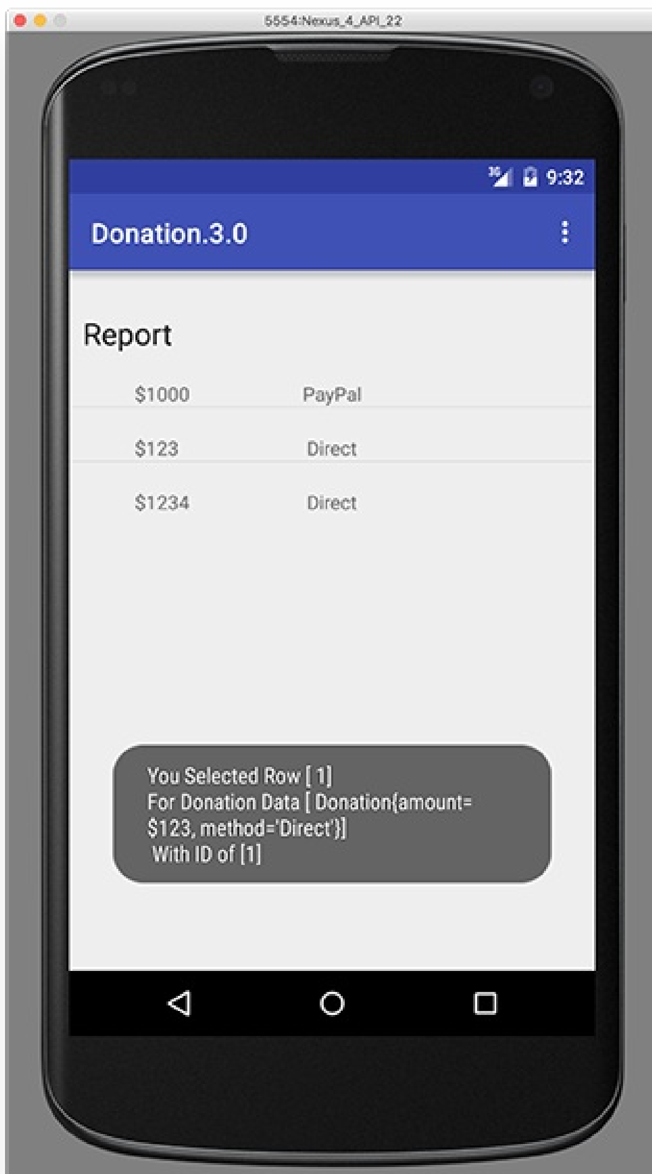
        listView = (ListView) findViewById(R.id.reportList);
        DonationAdapter adapter = new DonationAdapter(this,  don
ations);
        listView.setAdapter(adapter);
    }
}
```

Run your app once more, to confirm the changes and that your menu still works as it should.

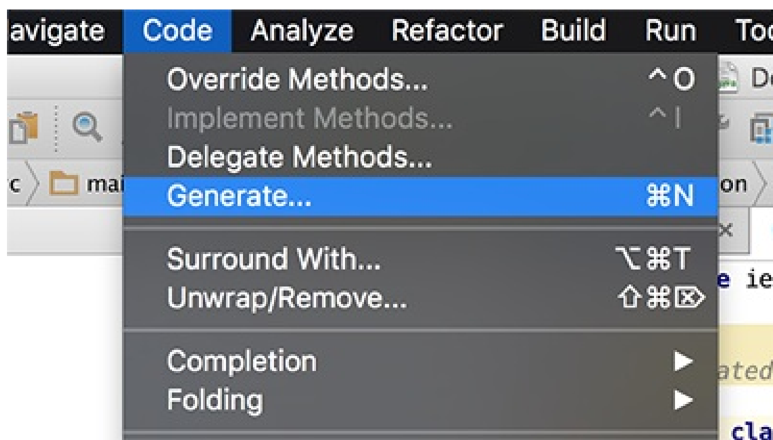
Exercise & Solution

As a final exercise, try and add some event handling to the Report Activity - i.e. when the user selects a row in Report List, display a simple toast detailing which row was selected and the donation data from that row.

Something like this:



You may find the following useful if you need to refactor your Donation model



and



Project Solution (and Starter for the next lab):

- Donation.4.0.Starter

Lab 05: Donation 4.0 - Database/Application Support

Objectives

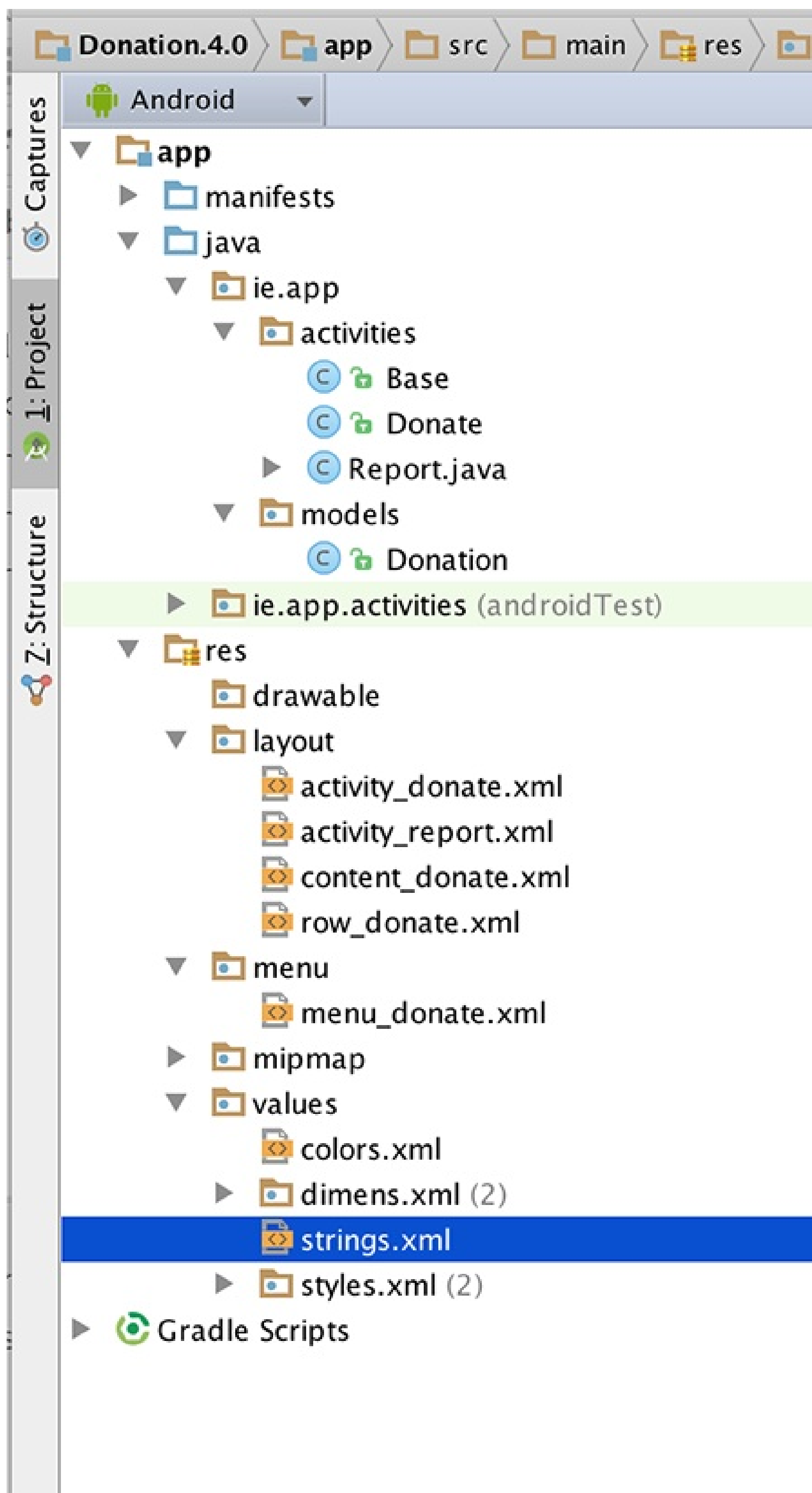
In this Lab, you will complete a final refactor of our Donation Case Study - **Donation.4.0**. We will build on the previous lab and add in some new features and Database Support, and an Application Object. On completion of this lab you'll be able to

- add Database Support to an Android Application
- work with an 'Application' object

Setup - Starter Code

As with the previous labs, you can download the solution/starter code for Donation.4.0.starter, or continue on with your own version.

Your current project (after renaming/copying) should looks as follows:



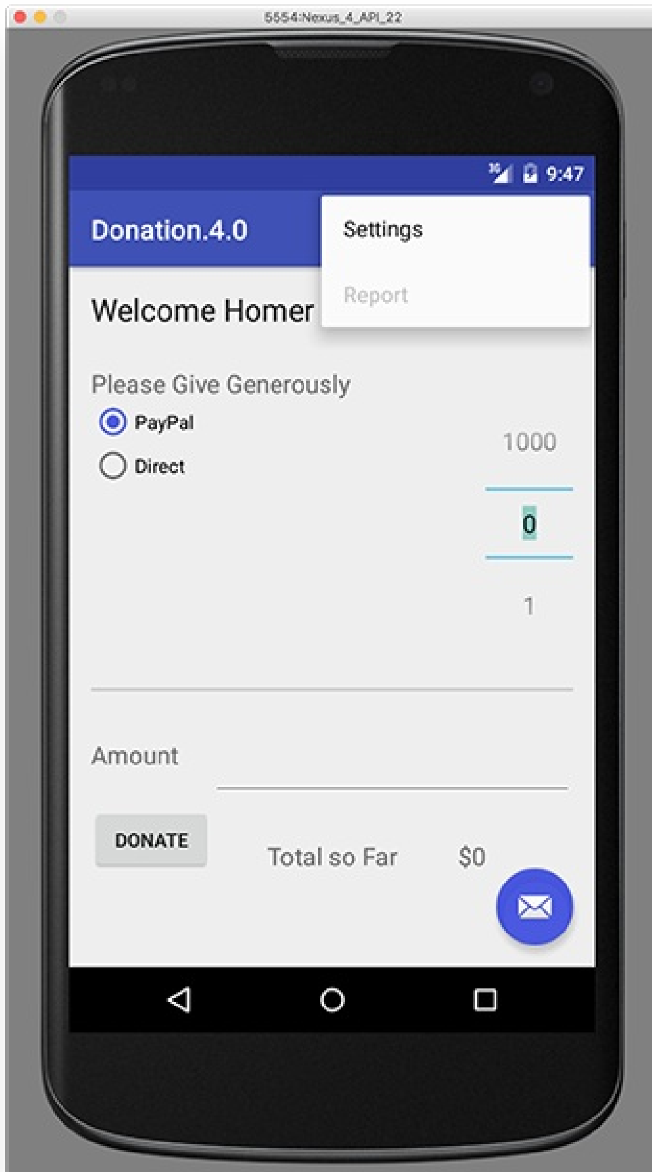
In this lab, you are required to do the following:

- Add a new Menu Option - 'Reset' - to clear out any donations after the target is reached
- Add Database Support to Donation to manage the donations made
- Refactor existing Classes to accommodate the new database classes

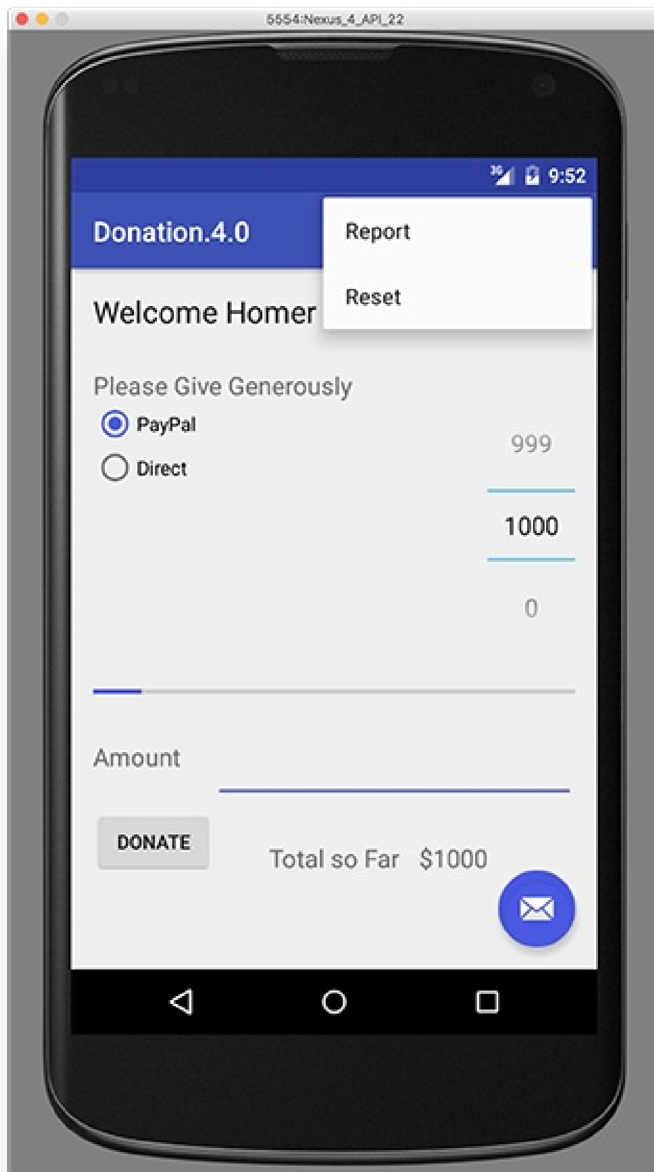
The following steps will guide you through these requirements, so we'll start with Menu Option.

Adding a new Menu Option

First of all, confirm that the current Menu looks like this:



but we want something like this:



The first thing to do is add in a new resource in strings.xml (or use Android Studio (Alt + Return) to fix the string resource error if you paste in the menu item directly)

```
<string name="menuReset">Reset</string>
```

and then the corresponding menu item in donate.xml

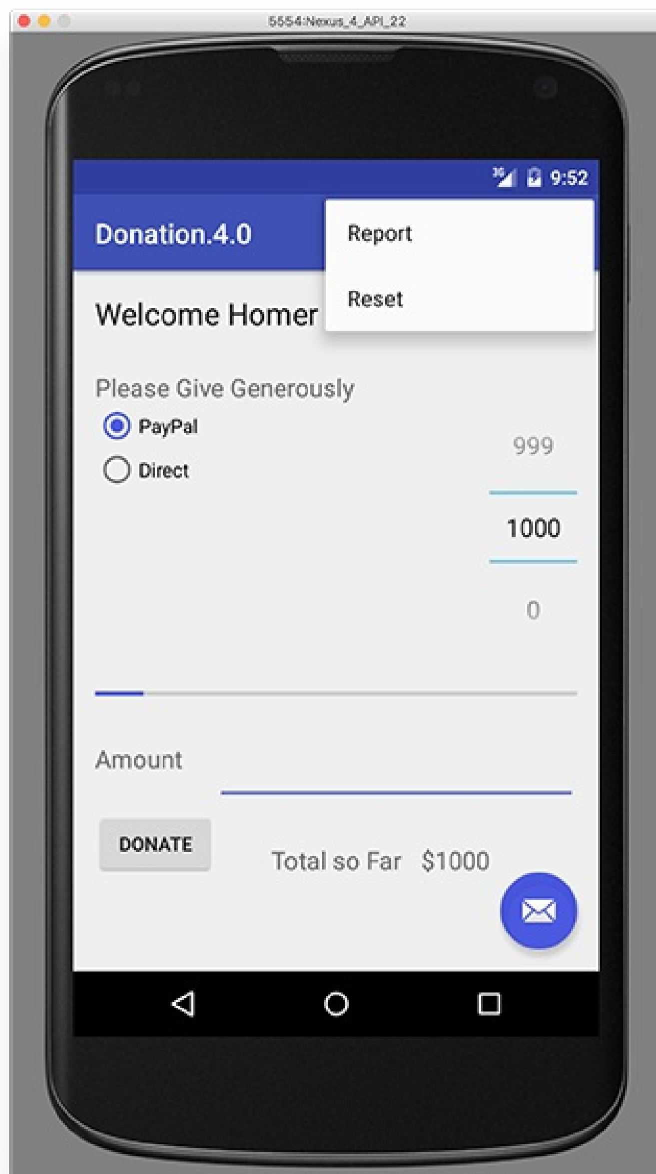
```
    <item
        android:id="@+id/menuReset"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:title="@string/menuReset"
        android:onClick="reset"/>
```

It's probably worth removing the 'Settings' menu item at this stage too, and its related method in the Base class. Next, edit **Base.java** and add in the following method stub

```
public void reset(MenuItem item) {}
```

to ensure our app won't crash when the menu loads (and looks for a method 'reset')

Run the app again and confirm you get the following Menu :



We can't implement this menu option fully yet, so for the moment, we'll just 'reset' the target amount back to zero (0) - Step 03.

Resetting the Target Amount

This is more of an interim step but is necessary to ensure the menu event handler for the 'Reset' option is working correctly.

First, edit **Donate.java** and introduce an implementation of the 'reset' method

```
 @Override
  public void reset(MenuItem item)
  {
        // Your implementation goes here
  }
```

the

```
@Override
```

annotation is important - can you explain why?

So add in the code necessary to deal with the Reset Menu option being selected, and reset the *totalDonated* back to zero (0). You also need to update the Donate UI to reflect this reset, so try and have a go at that too.

Run the app again to confirm that the 'Reset' Menu option is now functioning.

Application Object

Before we complete this step, here's the code you need for the previous step.

```
 @Override
  public void reset(MenuItem item)
  {
    totalDonated = 0;
    amountTotal.setText("$" + totalDonated);
    donations.clear();
  }
```

In order to keep out application design coherent, we now bring in an 'Application' object.

Create a new package called 'ie.app.main' and incorporate this class here:

```
package ie.app.main;

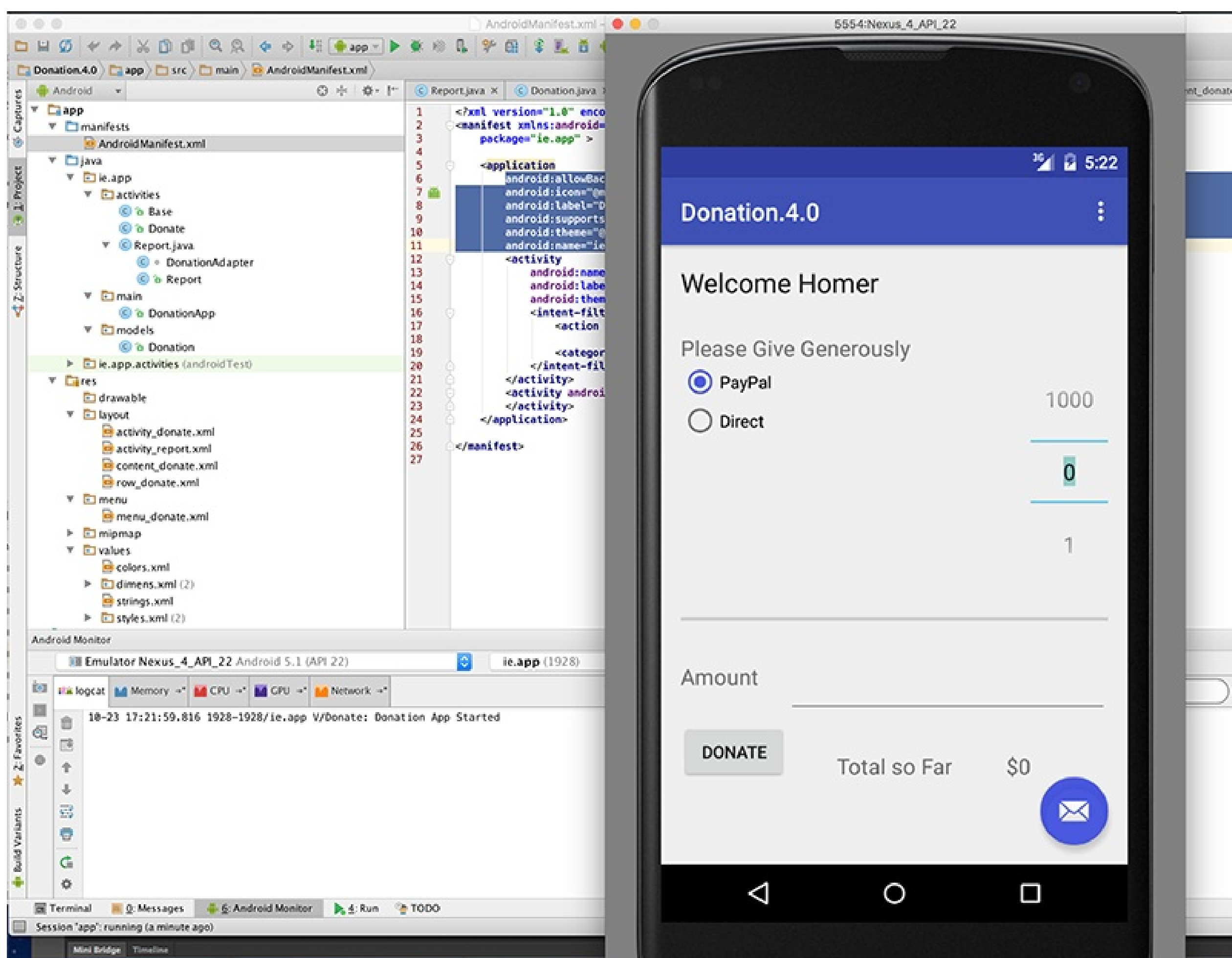
import android.app.Application;
import android.util.Log;

public class DonationApp extends Application
{
  @Override
  public void onCreate()
  {
    super.onCreate();
    Log.v("Donate", "Donation App Started");
  }
}
```

Application objects need to be references in the AndroidManifest.xml - at the very top as 'andorid:name'

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme"
    android:name="ie.app.main.DonationApp">
```

Make sure the 'Donation App Started' appears in the logs to verify that it has actually been engaged correctly, when you launch the app.



Donation Model

We now need to refactor the Base class (next Step) and move the donation related attributes and method (i.e. the variables target, totalDonated and the donations list,and the newDonation() method) into our DonationApp class.

This is a revised version of DonationApp - which now manages a list of donations. It also centralises the 'makeDonation' event implementing it as a method. Replace your donation with this one:

```
package ie.app.main;

import java.util.ArrayList;
import java.util.List;

import android.app.Application;
import android.util.Log;
import android.widget.Toast;
import ie.app.models.Donation;

public class DonationApp extends Application
{
  public final int        target       = 10000;
  public int              totalDonated = 0;
  public List <Donation> donations     = new ArrayList<Donation>(
);

  public boolean newDonation(Donation donation)
  {
    boolean targetAchieved = totalDonated > target;
    if (!targetAchieved)
    {
      donations.add(donation);
      totalDonated += donation.amount;
    }
    else
    {
      Toast.makeText(this, "Target Exceeded!", Toast.LENGTH_SHOR
```

```
T).show();
    }
    return targetAchieved;
  }

  @Override
  public void onCreate()
  {
    super.onCreate();
    Log.v("Donation", "Donation App Started");
  }
}
```

Base Class Refactoring

The Base activity can now be completely refactored to make use of the DonationApp object. You **WILL** have errors at the end of this step, due to referencing our (as yet missing) database classes - but we'll fix those in the next few steps.

This is our new Base class

```
package ie.app.activities;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.os.Bundle;

import ie.app.R;
import ie.app.main.DonationApp;

public class Base extends AppCompatActivity {

    public DonationApp app;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        app = (DonationApp) getApplication();

        app.dbManager.open();
        app.dbManager.setTotalDonated(this);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        app.dbManager.close();
```

```
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        getMenuInflater().inflate(R.menu.menu_donate, menu);
        return true;
    }

    @Override
    public boolean onPrepareOptionsMenu (Menu menu){
        super.onPrepareOptionsMenu(menu);
        MenuItem report = menu.findItem(R.id.menuReport);
        MenuItem donate = menu.findItem(R.id.menuDonate);
        MenuItem reset = menu.findItem(R.id.menuReset);

        if(app.dbManager.getAll().isEmpty())
        {
            report.setEnabled(false);
            reset.setEnabled(false);
        }
        else {
            report.setEnabled(true);
            reset.setEnabled(true);
        }
        if(this instanceof Donate){
            donate.setVisible(false);
            if(!app.dbManager.getAll().isEmpty())
            {
                report.setVisible(true);
                reset.setEnabled(true);
            }
        }
        else {
            report.setVisible(false);
            donate.setVisible(true);
            reset.setVisible(false);
        }

        return true;
```

```
    }

    public void report(MenuItem item)
    {
        startActivity (new Intent(this, Report.class));
    }

    public void donate(MenuItem item)
    {
        startActivity (new Intent(this, Donate.class));
    }

    public void reset(MenuItem item) {}
}
```

Adding Database Support

Once you import the necessary Database classes, (to fix the errors from the previous step) this step is relatively straight forward - all you have to do is replace the method calls that manages the ***donationList*** with the respective ***dbManager*** calls.

First thing to do is download the necessary database classes in the database archive and add them to a new *ie.app.database* package in your project.

Take a few moments to investigate the classes and familarise yourself with the methods you'll be using.

There are a few classes you'll need to modify to add database support to your project, but initially, you need to create an instance of ***DBManager*** in **Base.java** and both open/close the database when necessary, so refer to the Lecture Material to complete this.

Next, update your **DonationApp** class with the following:

```
public class DonationApp extends Application
{
    public final int       target       = 10000;
    public int             totalDonated = 0;
    //public List <Donation> donations    = new ArrayList<Donati
on>();
    public DBManager dbManager;


    public boolean newDonation(Donation donation)
    {
        boolean targetAchieved = totalDonated > target;
        if (!targetAchieved)
        {
            dbManager.add(donation);
            totalDonated += donation.amount;
        }
        else
        {
            Toast.makeText(this, "Target Exceeded!", Toast.LENGT
H_SHORT).show();
        }
        return targetAchieved;
    }

    @Override
    public void onCreate()
    {
        super.onCreate();
        Log.v("Donate", "Donation App Started");
        dbManager = new DBManager(this);
        Log.v("Donate", "Database Created");

    }
}
```

Note the references to a new *dbManager* object.

Also, our Donation class needs a slight refactor, so replace the current class with this one.

```
public class Donation
{
  public int    id;
  public int    amount;
  public String method;

  public Donation (int amount, String method)
  {
    this.amount = amount;
    this.method = method;
  }

  public Donation ()
  {
    this.amount = 0;
    this.method = "";
  }

  public String toString()
  {
    return id + ", " + amount + ", " + method;
  }
}
```

Once you make these changes, commenting out the donationList List, (and save the file) you'll get a number of errors, which actually indicates which classes you need to now update and add the database calls (and remove the donationList calls).

Each error requires only one line of code to be fixed, so have a go and updating each of the classes (and we'll have a look at the solution near the end of the Practical Lab).

Once you fix all the errors, and run the app again, you should still be able to make donations - but this time those donations are stored in a database.

And as a final check, if you shut down the app and restart it again, you should still be able to see the donations Report.

Resetting the Donations

The last step in this lab involves deleting all the donations in the database when the user wishes to 'Reset'.

There's actually not a lot required in this step - all you need to do is call **reset()** on your datbase object when the user selects the Menu option, so modify your reset method (in your Donate.java) as follows:

```
@Override
public void reset(MenuItem item)
{
        app.dbManager.reset();
      app.totalDonated = 0;
      amountTotal.setText("$" + app.totalDonated);
}
```

You also need to update your *onPrepareOptionsMenu()* method in your **Base** class to handle the 'Reset' menu option being disabled/displayed properly, so refer to the lecture material for this.

That's about it - with one exception. There's a small bug in the app related to when the app restarts and the target **HAS NOT** been reached.

Can you find it, and more importantly, fix it?

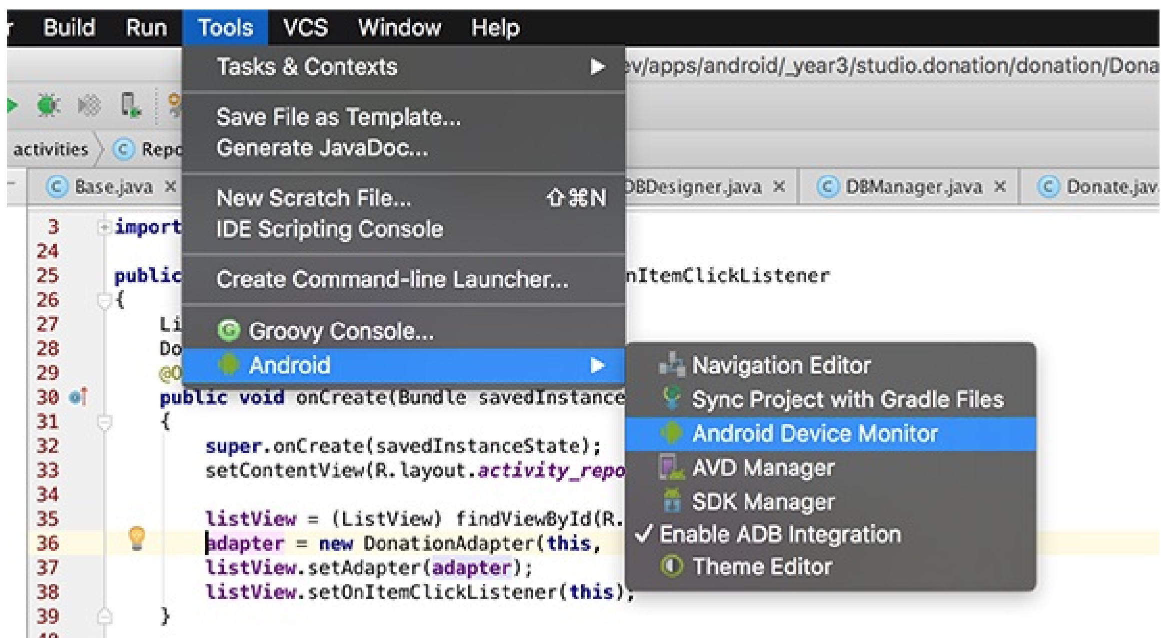
Android Device Monitor

As an exercise of sorts, you should become familar with the Android Device Monitor, particularly how it relates to viewing your database on the emulator/device.

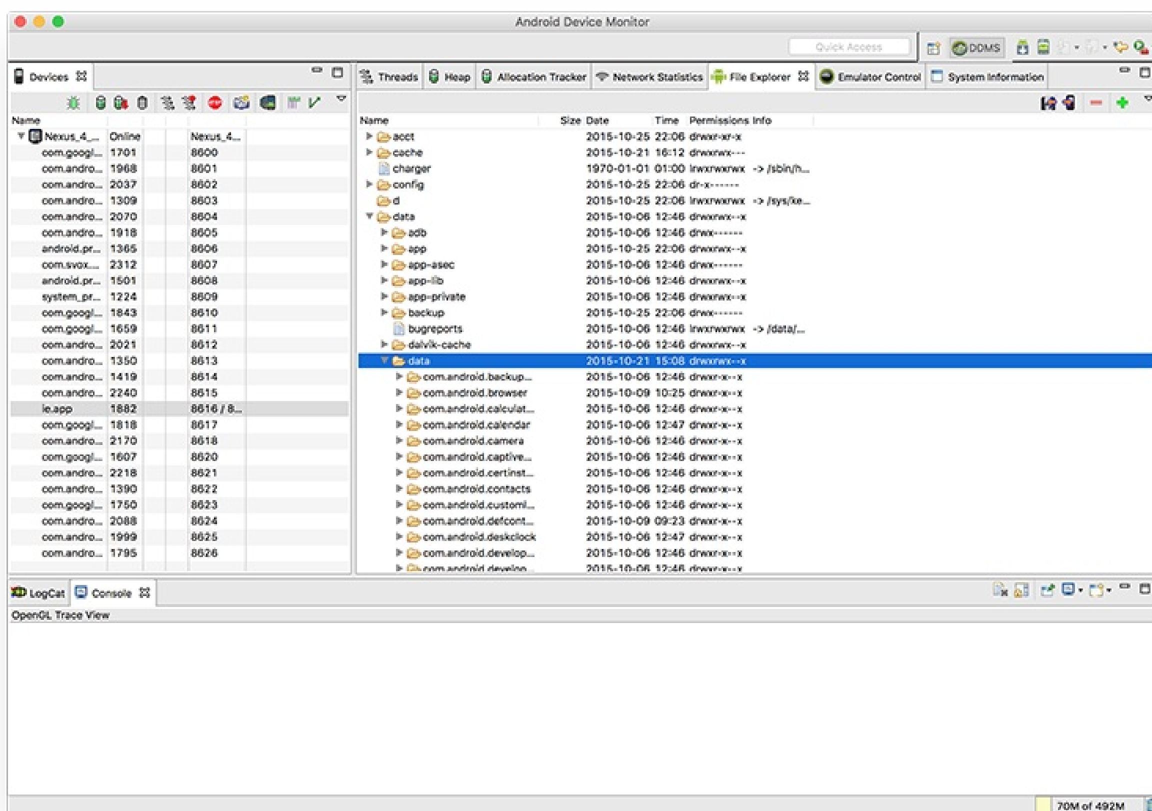


In Android Studio, you launch the Android Device Monitor as follows:

Tools->Android->Android Device Monitor (as below)



Next, you need to navigate to the data/data folder within the application package on the device (in our case ie.app) like so:



Then, scroll/find your database (donations.db) in the ie.app.databases folder:

Step 09

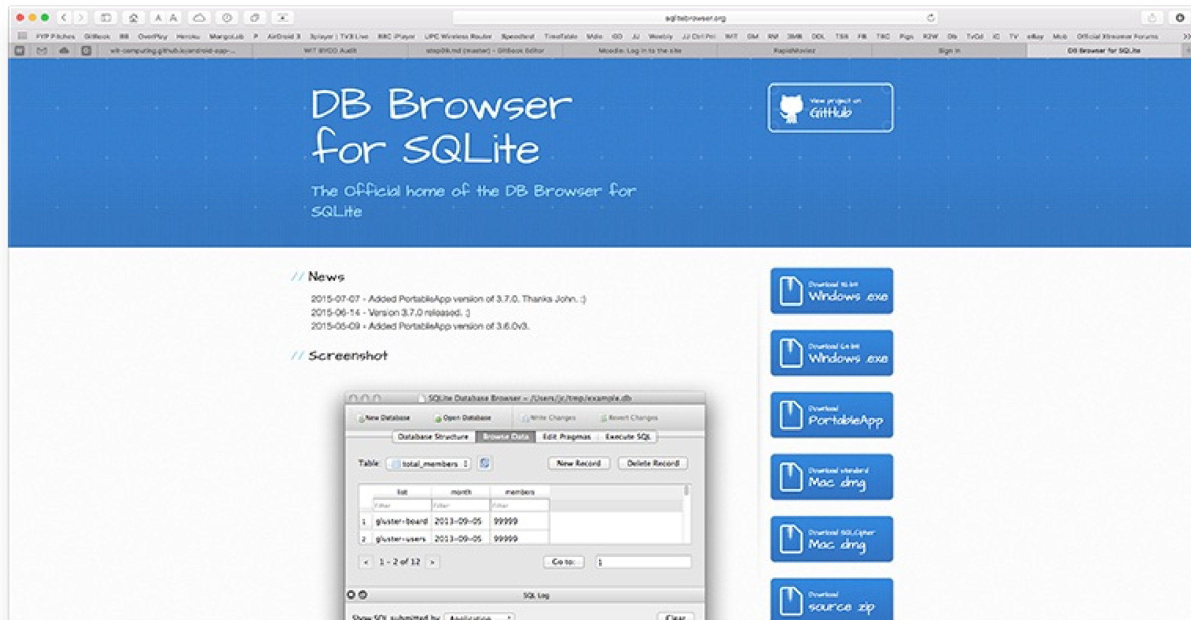


Select the "Pull a file from the Device" on the top right-hand-side of the window:

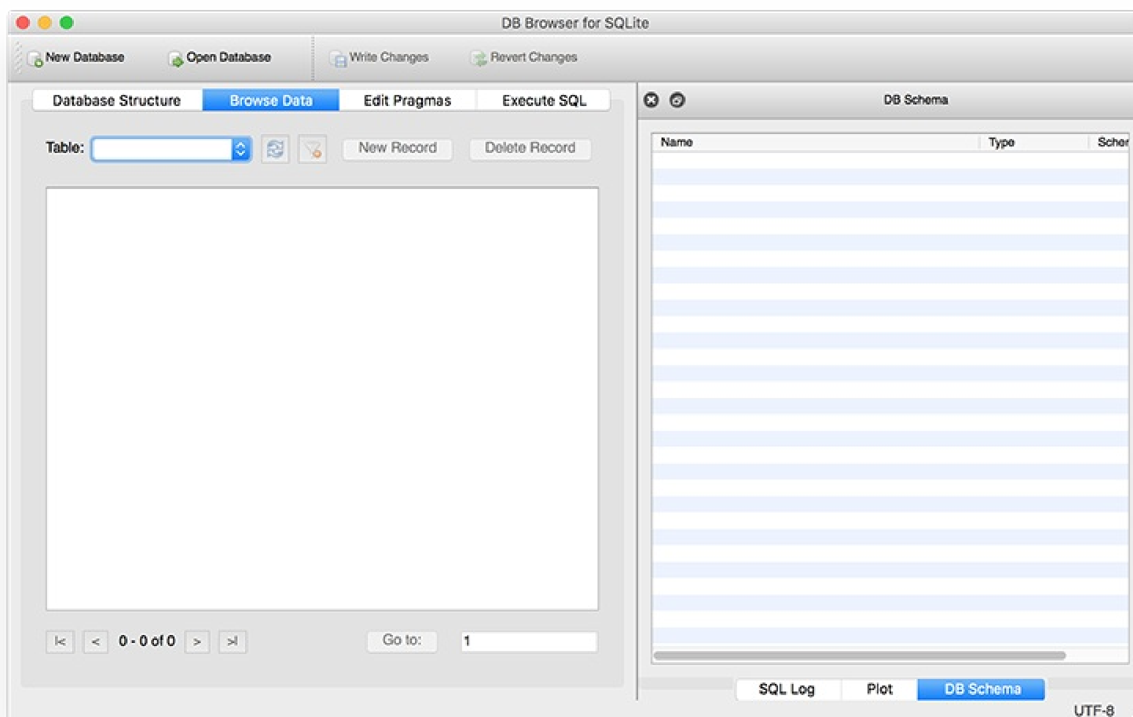


and save your database file to a local folder.

Next download and install sqlitebrowser which will allow us to view our database graphically.

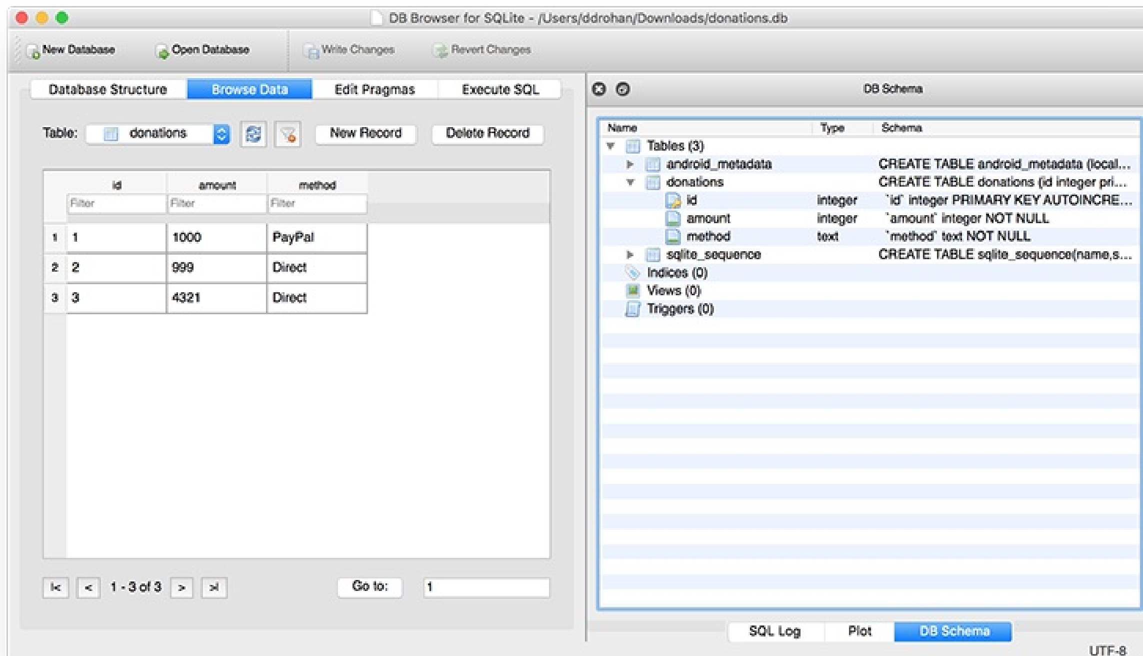


Launch your sqlitebrowser app/program to get this window:

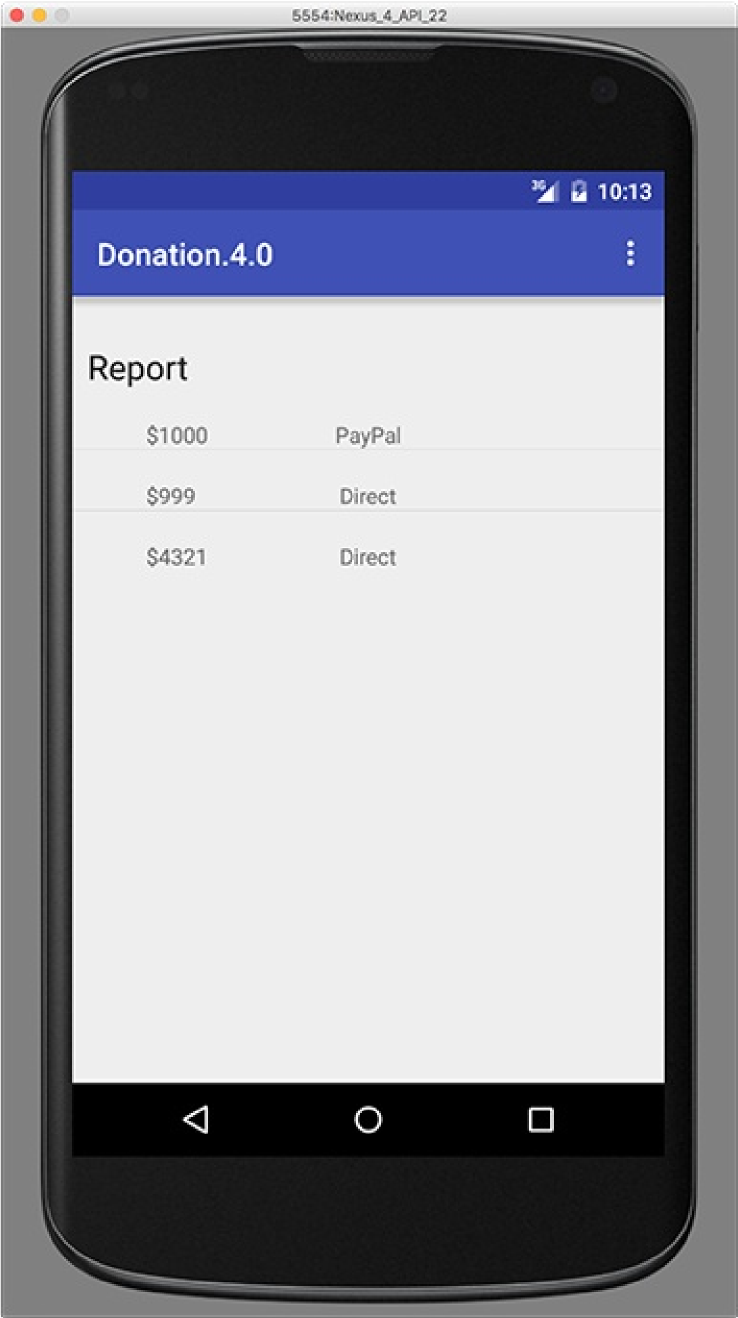


and then 'Open Database' selecting the database you pulled from the device. If everything goes to plan, you should be able to view your database table(s) and their content, as well as the SQL that created the tables.

Below, we can see a donations table with 3 donations



and the corresponding entries in our Android App.


5554:Nexus_4_API_22
10:13
Donation.4.0
Report
$1000
PayPal
$999
Direct
$4321
Direct

Solution

This is a solution to the lab:

- Donation.4.0

Lab 06: Donation 5.0 - REST/Cloud Support

Objectives

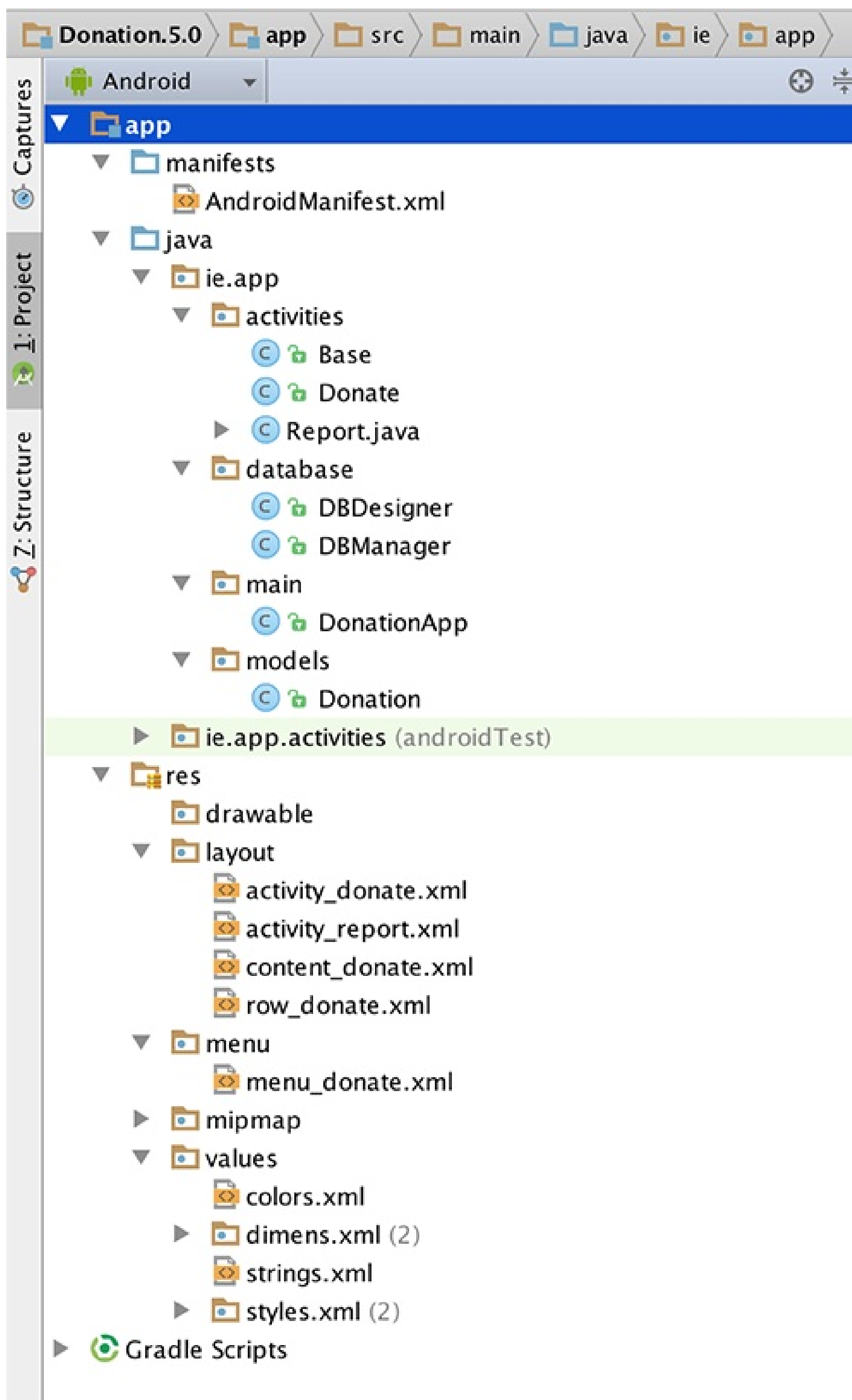
In this Lab, you will complete an additional refactor of our Donation Case Study - **Donation.5.0**. We will build on the previous lab and add in some new features and most importantly,connect to a REST based Web Service. On completion of this lab you'll be able to

- connect to a remote server to retrieve & delete Donations
- connect to a remote server to retrieve a single Donation
- add Event Handling to an Image on our Custom Donation Row

Setup - Starter Code

As with the previous labs, you can download the solution/starter code for Donation.5.0.starter, or continue on with your own version.

Your current project (after renaming/copying) should looks as follows:



In this lab, you are required to do the following:

- Add REST Support via the Donation sister site
- Remove the Database Support and revert to the original Donation List
- Refactor existing Classes to accommodate the new REST API

The following steps will guide you through these requirements, so we'll start with bringing in the classes we need to connect to our Web Service.

Adding REST Support & Project Clean Up

The main purpose of this version of Donation is to connect to a Web Service (our sister site) and be able to retrieve, insert and delete Donations.

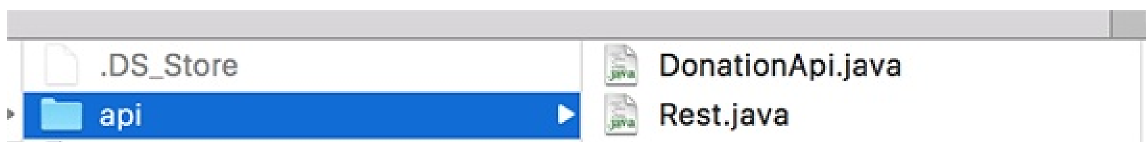
To make things a bit easier I've devloped a simple API to make the HTTP calls and convert the responses from JSON into objects our Android App can use.

So go ahead and download the package here.

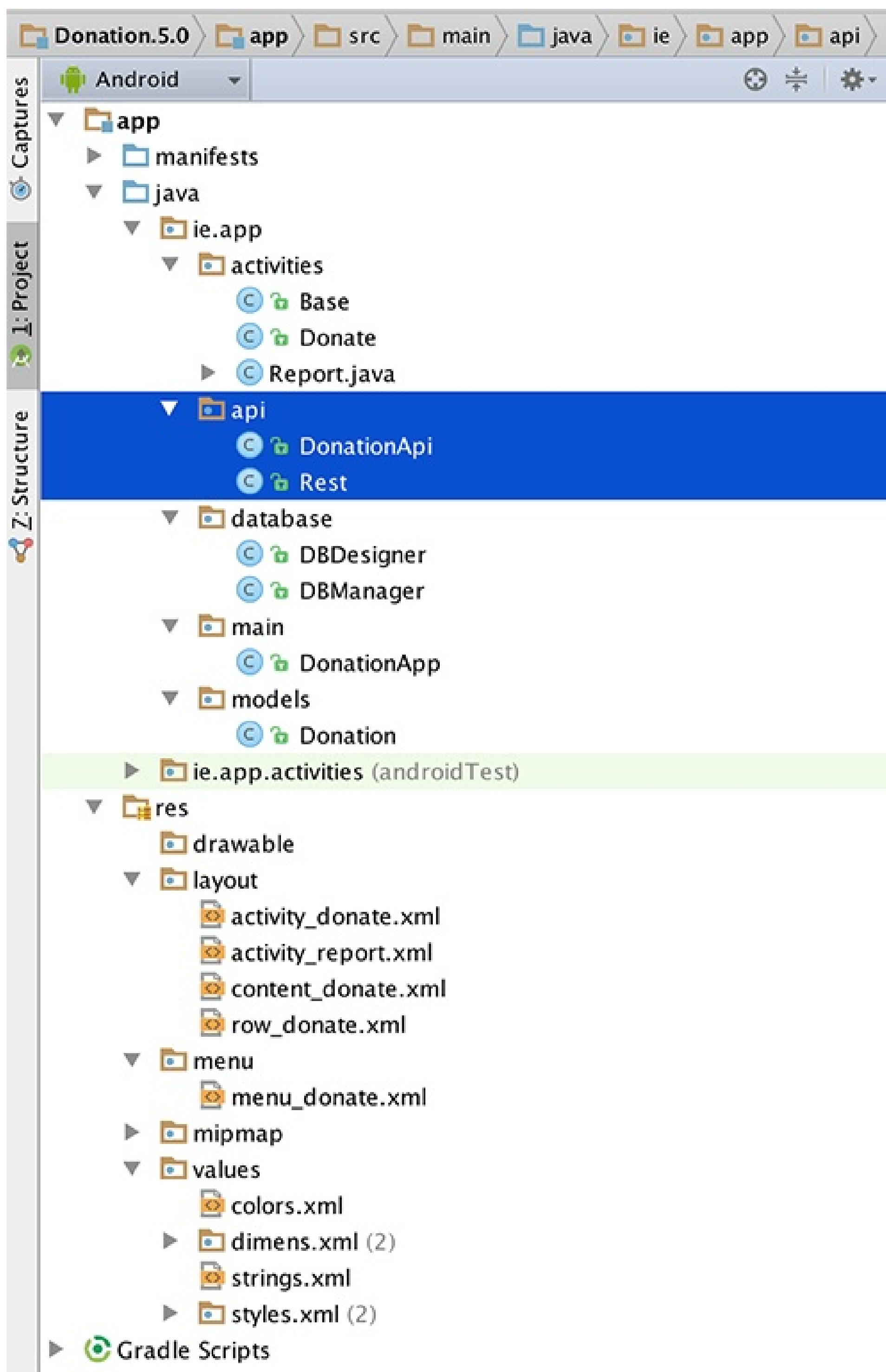
The extracted archive consists of the following:

- DonationApi.java
- Rest.java

(viewed in Finder)



You will need to add these classes to your Android Studio Project and the simplest way is to copy the api folder in Windows Explorer or Finder and paste directly into your **ie.app** package in your Android Studio Project. Once completed, your project should look something like this:

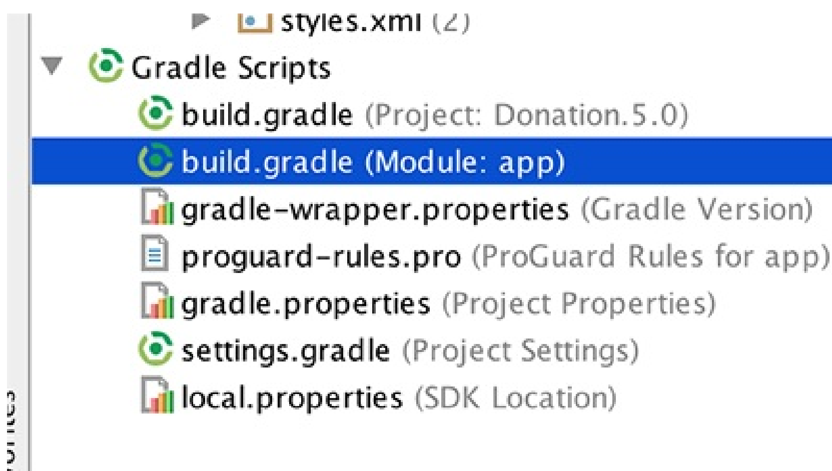


Now, if you Rebuild the project

Build->Rebuild Project

you'll have a number of errors relating to Googles Gson, so first thing to do is add **Googles Gson's** dependency to our project:

Open your 'build.gradle' file for **app**, **NOT THE PROJECT!**



Add the Gson dependency

```
compile 'com.google.code.gson:gson:2.2.3'
```

so your dependencies in your build file looks something like this

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.0.1'
    compile 'com.android.support:design:23.0.1'

    compile 'com.google.code.gson:gson:2.2.3'
}
```

Rebuild your project again and you should now be error free!

The next thing we need to do is remove/delete our database classes and reintroduce a simple list back into our DonationApp Application class.

So,

- first, just delete the whole **ie.app.database** package (don't worry about the errors, we'll fix those soon)
- next, reintroduce our donations list into the DonationApp class

```
public List <Donation> donations    = new ArrayList<Donation
>();
```

- finally, remove all references to **dbManager** in the project and replace with our **donations** or **app.donations** (depending on the context).

We've actually taken a step back as regards functionality, as we're not adding donations to our list directly - the list is only to be used to hold the result of our REST calls.

So let's go ahead and start using our REST classes for data retrieval.

Donate Activity - Getting the Running Total

When our Donation App initially starts we want to ensure that the current total (if any) is set correctly and corresponds to the donations listed on our sister web app.

We need to retrieve a list of all our donations from the Server and set our total. We will achieve this through our REST classes and the use of AsyncTasks to execute those calls on a background thread.

Before we start, we need to allow our app to access the internet (and network) so we need to add some permissions to our manifest file, so add the following to your **AndroidManifest.xml** file (just before the <application tag)

```
    <uses-permission android:name="android.permission.INTERNET"/>

    <uses-permission android:name="android.permission.ACCESS_NET
WORK_STATE" />
```

Introduce the following **private** class into the Donate activity

```
private class GetAllTask extends AsyncTask<String, Void, List<Do
nation>> {

        protected ProgressDialog         dialog;
        protected Context                 context;

        public GetAllTask(Context context)
        {
            this.context = context;
        }

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            this.dialog = new ProgressDialog(context, 1);
```

```
            this.dialog.setMessage("Retrieving Donations List");
            this.dialog.show();
        }

        @Override
        protected List<Donation> doInBackground(String... params)
 {
            try {
                Log.v("donate", "Donation App Getting All Donati
ons");
                return (List<Donation>) DonationApi.getAll((Stri
ng) params[0]);
            }
            catch (Exception e) {
                Log.v("donate", "ERROR : " + e);
                e.printStackTrace();
            }
            return null;
        }

        @Override
        protected void onPostExecute(List<Donation> result) {
            super.onPostExecute(result);

            //use result to calculate the totalDonated amount he
re

            progressBar.setProgress(app.totalDonated);
            amountTotal.setText("$" + app.totalDonated);

            if (dialog.isShowing())
                dialog.dismiss();
        }
    }
```

To actually invoke this Task, add the following method

```
@Override
    public void onResume() {
        super.onResume();
        new GetAllTask(this).execute("/donations");
    }
```

If you try and run your app, it'll connect to the web service and return our list from the sister site - but it will then crash. Can you work out why this is happening from the Logs, and how to fix it?

Refactoring our Donation Model

If you've payed particular attention to the JSON string that was returned in the previous request, you'll see that there are 4 attributes, of different types - which don't match our current **Donation** class.

This is why the app crashed - we were trying to convert our json string into a list of objects that are not the same type.

So go ahead and replace your current Donation Model with this one

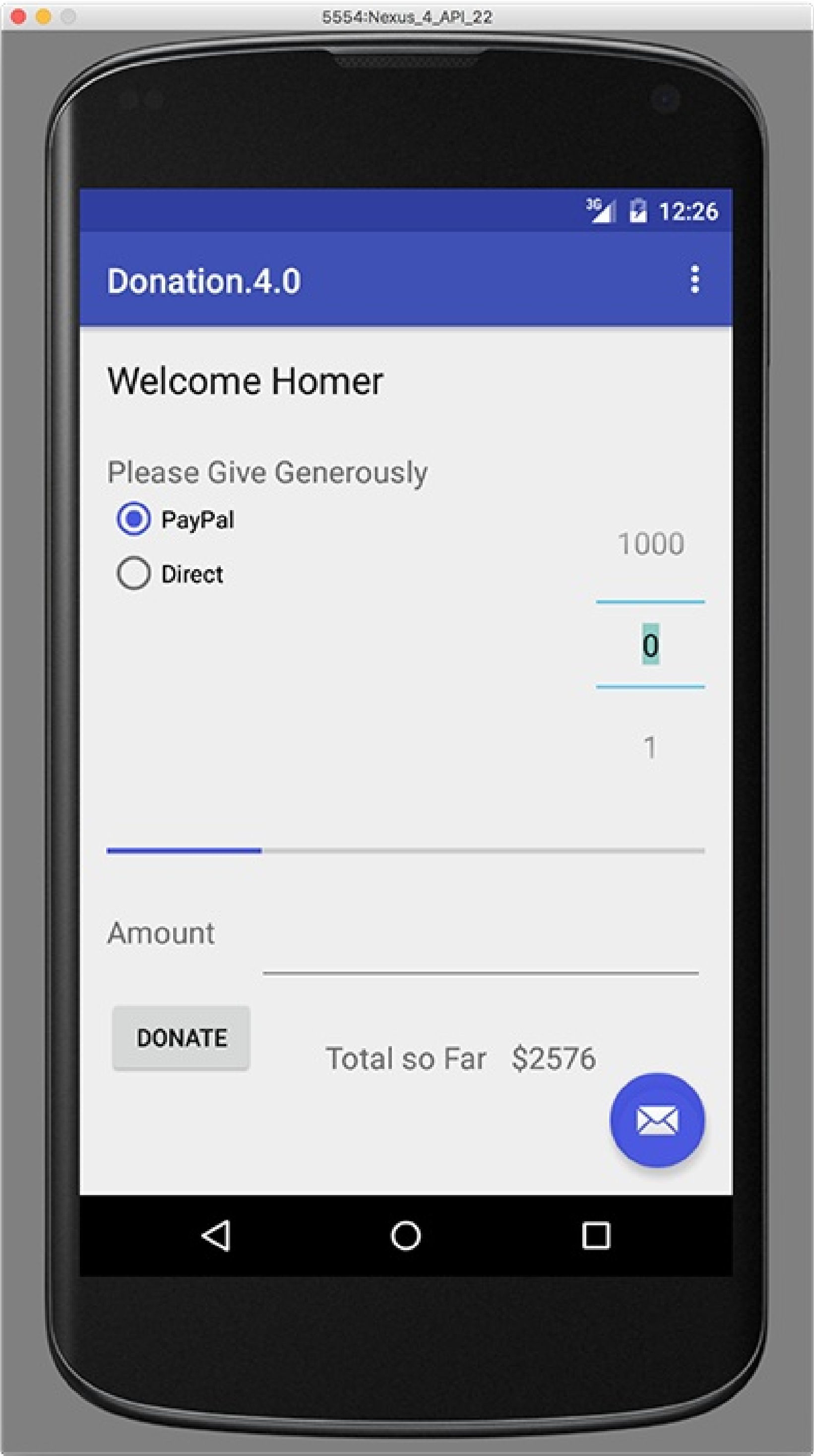
```
public class Donation
{
    public String _id;
    public int    amount;
    public String paymenttype;
    public int    upvotes;

    public Donation (int amount, String method, int upvotes)
    {
        this.amount = amount;
        this.paymenttype = method;
        this.upvotes = upvotes;
    }

    public Donation ()
    {
        this.amount = 0;
        this.paymenttype = "";
        this.upvotes = 0;
    }

    public String toString()
    {
        return _id + ", " + amount + ", " + paymenttype + ", " +
 upvotes;
    }
}
```

There'll be a few small errors to fix, but once you do, run your app again and you should get something like the following:


5554:Nexus_4_API_22
12:26
Donation.4.0
Welcome Homer
Please Give Generously
PayPal
Direct
1000
0
1
Amount
DONATE
Total so Far $2576

Report Activity - Refactoring our Layouts

Now that we can see our Total Donations, let's display them in our Report Activity.

We've a few modifications to make on our layouts first, and we're going to introduce Androids SwipereRreshLayout feature so replace your current report layout with this one

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/re
s/android"
    android:orientation="vertical" android:layout_width="match_p
arent"
    android:layout_height="match_parent">


    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarg
e"
        android:text="@string/reportTitle"
        android:id="@+id/reportTitle"
        android:layout_marginLeft="0dp"
        android:layout_marginTop="31dp"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:paddingLeft="10dp" />

    <LinearLayout xmlns:android="http://schemas.android.com/apk/
res/android"
        android:orientation="horizontal" android:layout_width="f
ill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/reportTitle"
        android:layout_alignParentStart="true"
        android:id="@+id/linearLayout">
```

```
        <TextView
            android:id="@+id/row_upvotes"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="15dp"
            android:layout_marginTop="20dp"
            android:text="Upvotes"
            android:layout_weight="0.5"
            android:textSize="20sp"
            android:textColor="@color/colorPrimaryDark" />

        <TextView
            android:id="@+id/row_amount"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:text="Amount"
            android:layout_weight="0.5"
            android:textSize="20sp"
            android:textColor="@color/colorPrimaryDark" />

        <TextView
            android:id="@+id/row_method"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:text="Method"
            android:layout_weight="1.5"
            android:textSize="20sp"
            android:textColor="@color/colorPrimaryDark" />

    </LinearLayout>

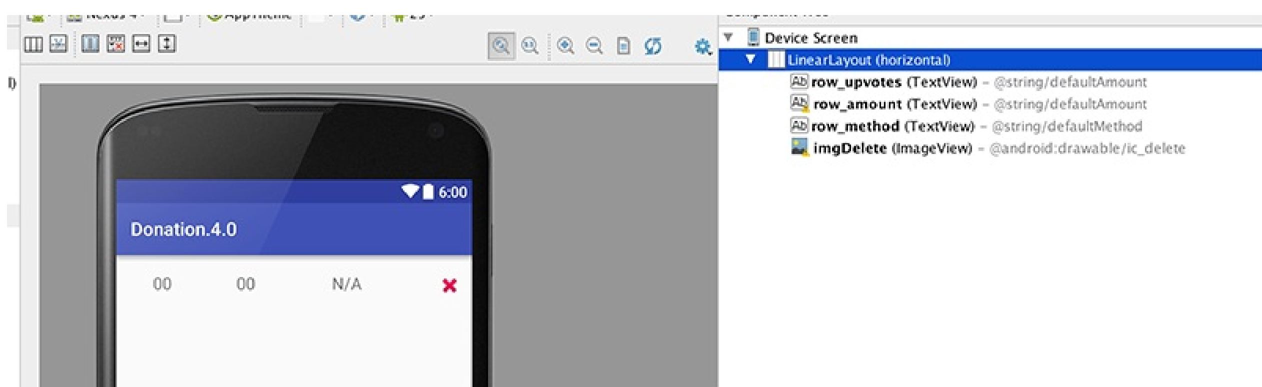
    <android.support.v4.widget.SwipeRefreshLayout
        android:id="@+id/report_swipe_refresh_layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/linearLayout">
```

```
        <ListView
            android:id="@+id/reportList"
            android:layout_width="match_parent"
            android:layout_height="match_parent">
        </ListView>

    </android.support.v4.widget.SwipeRefreshLayout>

</RelativeLayout>
```

We also need to modify our custom row, so the user can see the number of 'upvotes' for each donation, so go ahead and refactor your row_donate.xml by introducing a new TextView Resource to display the 'upvotes'. We're also going to be able to delete a donation via a 'delete' button, so while you're refactoring the custom row, have a go at that too (like so)



(And don't worry, the solution is on the next step, but have a go first)

Report Activity - Displaying our Donations

Here's the updated custom layout you should have (or something similar)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    android:orientation="horizontal" android:layout_width="fill_
parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/row_upvotes"
        android:layout_width="80dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="40dp"
        android:layout_marginTop="20dp"
        android:text="@string/defaultAmount"
        android:textSize="18sp" />

    <TextView
        android:id="@+id/row_amount"
        android:layout_width="80dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:layout_marginLeft="10dp"
        android:text="@string/defaultAmount"
        android:layout_weight="1"
        android:textSize="18sp" />

    <TextView
        android:id="@+id/row_method"
        android:layout_width="80dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"

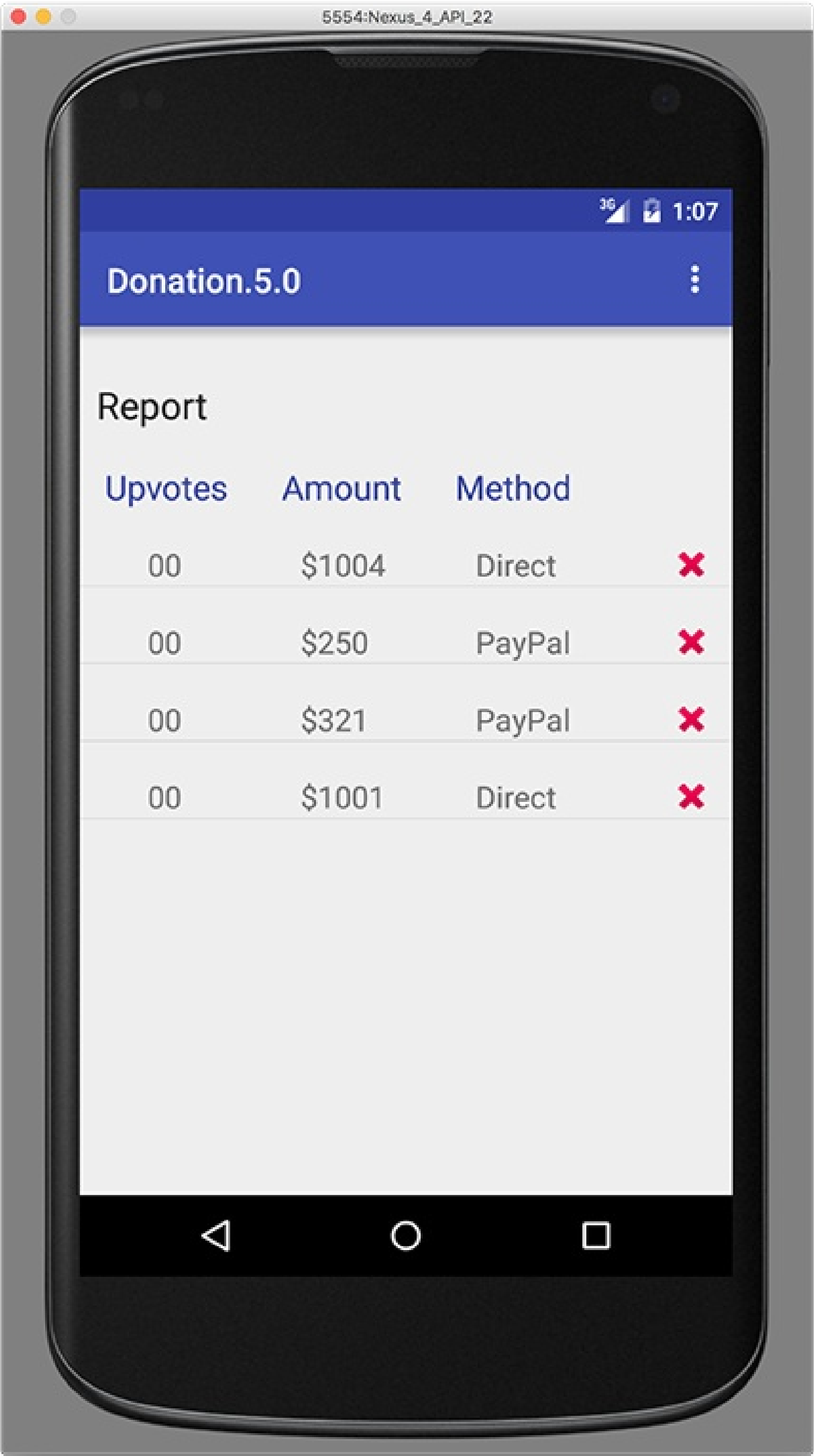
        android:text="@string/defaultMethod"
        android:layout_weight="1"
```

```
        android:textSize="18sp" />

    <ImageView
        android:id="@+id/imgDelete"
        android:layout_width="25dp"
        android:layout_height="25dp"
        android:layout_marginTop="20dp"
        android:layout_weight="1"
        android:src="@android:drawable/ic_delete" />

</LinearLayout>
```

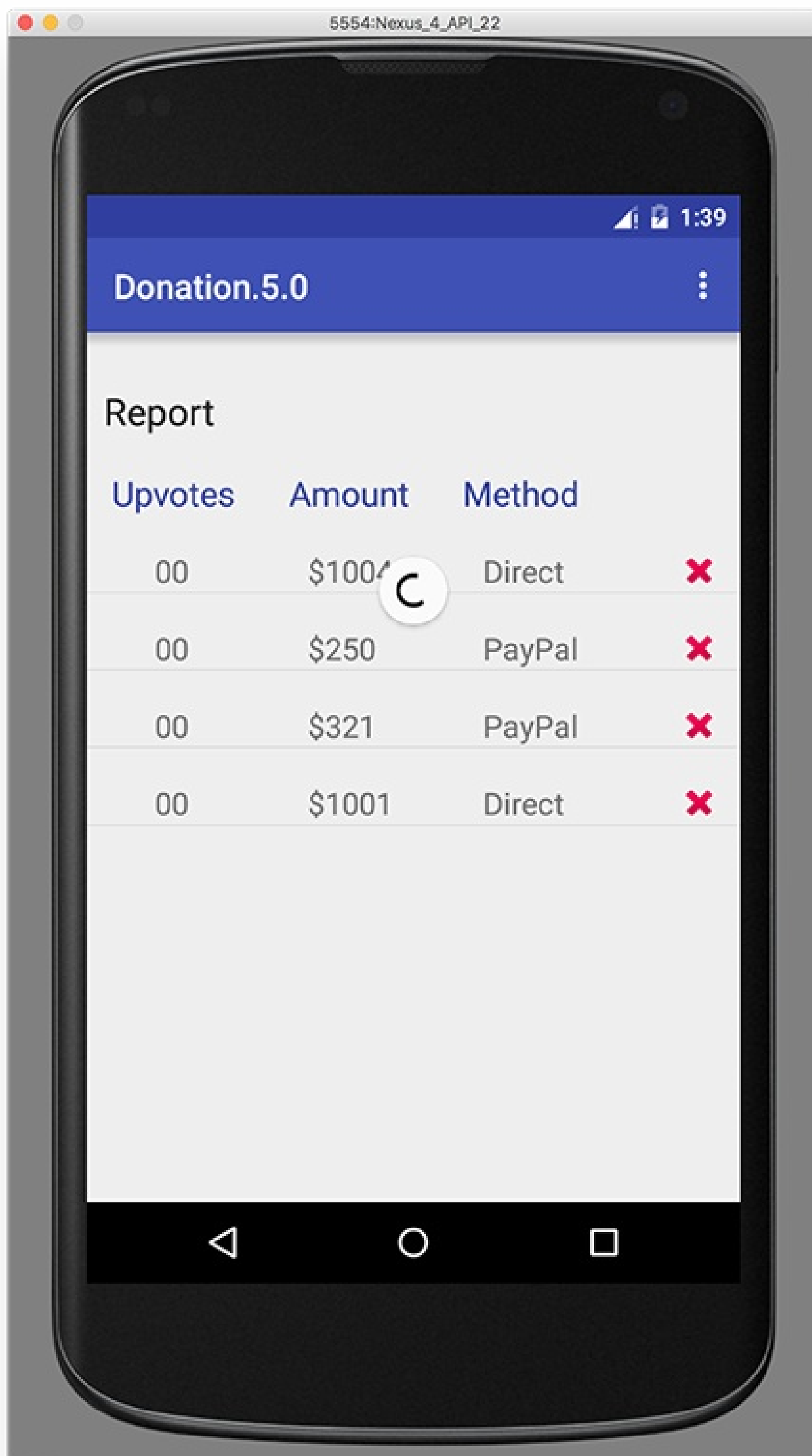
Now, if you run the app again, and select the Report Menu Option, you'll actually see the list that has been retrieved from the server (without any upvotes yet) - with no extra coding - how is this possible?



Experiment with the Swipe Refresh gesture and see what happens?

Report Activity - Implementing Swipe Refresh

You've probably noticed that even though we can 'refresh' our report list, all we see is the refresh progress spin indefinitely - we've no way to stop it except closing the activity.



The reason being, we have no backend implementation of the gesture to actually refresh the data in the list - so let's do that.

The first thing we need to do is introduce a new AsyncTask to retrieve the donations - more advanced approaches could utilise interfaces and inheritance to reuse already existing AsyncTasks (like our GetAllTask), but for the moment, we'll

try and keep it as simple as possible and write a new GetAllTask specifically for the Report activity.

Add the below AsyncTask to the Report Activity;

```
private class GetAllTask extends AsyncTask<String, Void, List<Do
nation>> {

        protected ProgressDialog dialog;
        protected Context context;

        public GetAllTask(Context context) {
            this.context = context;
        }

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            this.dialog = new ProgressDialog(context, 1);
            this.dialog.setMessage("Retrieving Donations List");
            this.dialog.show();
        }

        @Override
        protected List<Donation> doInBackground(String... params)
 {

            try {
                return (List<Donation>) DonationApi.getAll((Stri
ng) params[0]);
            } catch (Exception e) {
                Log.v("ASYNC", "ERROR : " + e);
                e.printStackTrace();
            }
            return null;
        }

        @Override
        protected void onPostExecute(List<Donation> result) {
            super.onPostExecute(result);
```

```
            app.donations = result;
            adapter = new DonationAdapter(context, app.donations
);
            listView.setAdapter(adapter);
            listView.setOnItemClickListener(Report.this);
            mSwipeRefreshLayout.setRefreshing(false);

            if (dialog.isShowing())
                dialog.dismiss();
        }
    }
```

Fix all the errors (bar 1) and then bring in the following reference

```
SwipeRefreshLayout mSwipeRefreshLayout;
```

replace the existing onCreate() method with this one

```
@Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_report);

        listView = (ListView) findViewById(R.id.reportList);
        mSwipeRefreshLayout = (SwipeRefreshLayout) findViewById(
R.id.report_swipe_refresh_layout);

        new GetAllTask(this).execute("/donations");

        mSwipeRefreshLayout.setOnRefreshListener(new SwipeRefres
hLayout.OnRefreshListener() {
            @Override
            public void onRefresh() {
                new GetAllTask(Report.this).execute("/donations"
);
            }
        });
    }
```

Run your app again and confirm that the refresh is working correctly, by adding or deleting a donation on the Web and then refreshing your Report Screen.

Note : as everyone and anyone :-) can add and/or delete donations via the web app, I'd encourage you to keep the list of donations to 4 or 5 while you're testing

We should also be displaying the 'upvotes' values for each donation so see if you can get that implemented before looking at the solution on the next step.

Report Activity - Event Handling

Here's the solution to our updated DonationAdapter class so replace your current DonationAdapter with this one, and run and test your app again, to complete the step.

```
class DonationAdapter extends ArrayAdapter<Donation> {
    private Context context;
    public List<Donation> donations;

    public DonationAdapter(Context context, List<Donation> donat
ions) {
        super(context, R.layout.row_donate, donations);
        this.context = context;
        this.donations = donations;
    }

    @Override
    public View getView(int position, View convertView, ViewGrou
p parent) {
        LayoutInflater inflater = (LayoutInflater) context
                .getSystemService(Context.LAYOUT_INFLATER_SERVIC
E);

        View view = inflater.inflate(R.layout.row_donate, parent
, false);
        Donation donation = donations.get(position);

        TextView amountView = (TextView) view.findViewById(R.id.
row_amount);
        TextView methodView = (TextView) view.findViewById(R.id.
row_method);
        TextView upvotesView = (TextView) view.findViewById(R.id
.row_upvotes);

        amountView.setText("" + donation.amount);
        methodView.setText(donation.paymenttype);
```

```
        upvotesView.setText("" + donation.upvotes);

        view.setTag(donation._id); // setting the 'row' id to th
e id of the donation

        return view;
    }

    @Override
    public int getCount() {
        return donations.size();
    }
}
```

While we working on the Report Activity, let's add some event handling so we can

- view the details of an individual Donation and
- delete a specific Donation via the delete Button

Introduce this AsyncTask into the Report Activity

```
private class GetTask extends AsyncTask<String, Void, Donation>
{

        protected ProgressDialog dialog;
        protected Context context;

        public GetTask(Context context) {
            this.context = context;
        }

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            this.dialog = new ProgressDialog(context, 1);
            this.dialog.setMessage("Retrieving Donation Details"
);
            this.dialog.show();
        }
```

```
        @Override
        protected Donation doInBackground(String... params) {

            try {
                return (Donation) DonationApi.get((String) param
s[0], (String) params[1]);
            } catch (Exception e) {
                Log.v("donate", "ERROR : " + e);
                e.printStackTrace();
            }
            return null;
        }

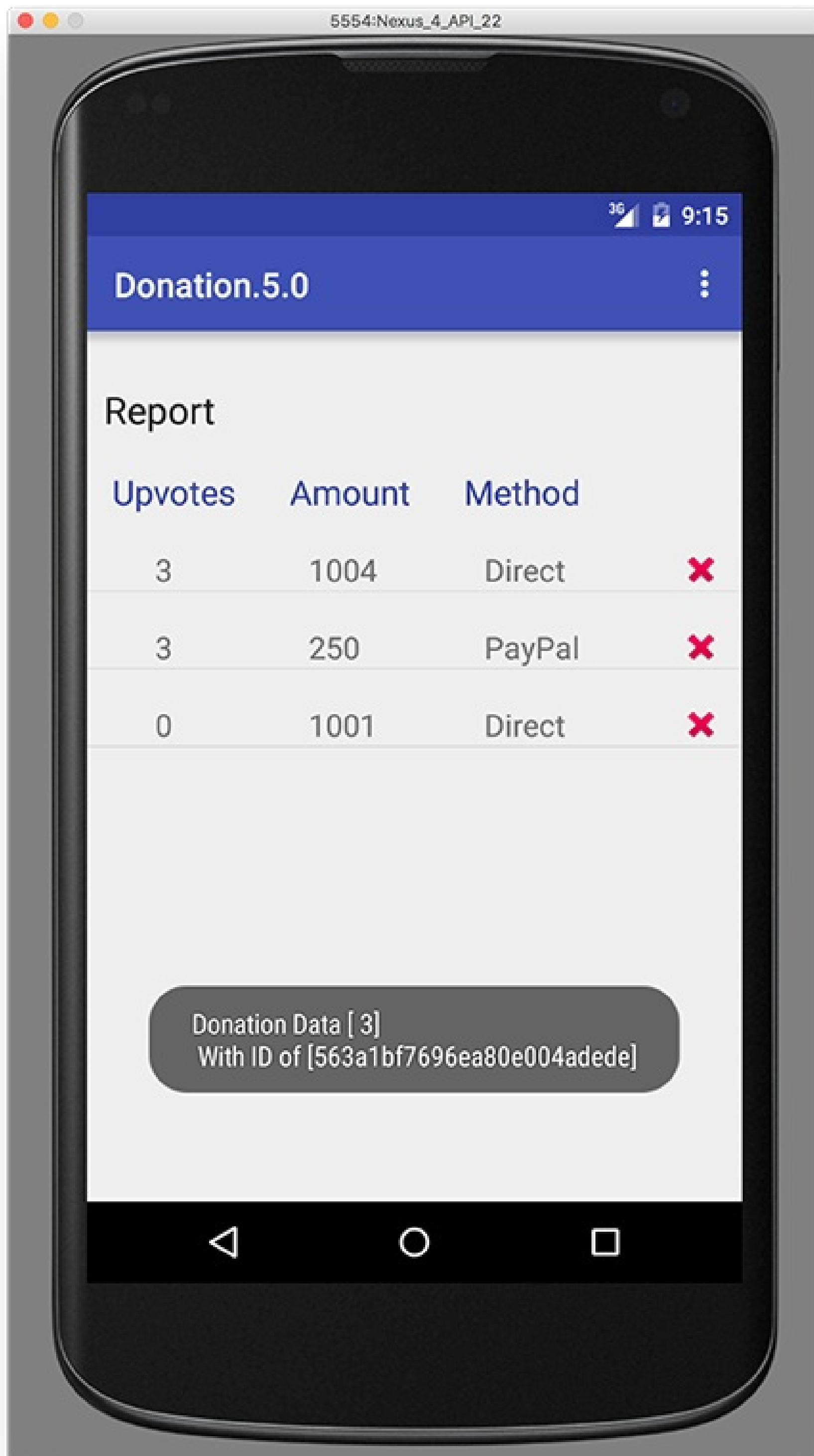
        @Override
        protected void onPostExecute(Donation result) {
            super.onPostExecute(result);

            Donation donation = result;

            Toast.makeText(Report.this, "Donation Data [ " + don
ation.upvotes + "]\n " +
                    "With ID of [" + donation._id + "]", Toast.L
ENGTH_LONG).show();

            if (dialog.isShowing())
                dialog.dismiss();
        }
    }
```

Fix any errors, and see can you add an **OnItemClickListener** to our Report class to display the Toast (below) when a row is selected, by invoking the above AsyncTask.



To implement the delete feature, we need to implement an **OnClickListener** interface so go ahead and do that and bring in this AsyncTask to begin with.

```
private class DeleteTask extends AsyncTask<String, Void, String>
{
```

```
        protected ProgressDialog dialog;
        protected Context context;

        public DeleteTask(Context context) {
            this.context = context;
        }

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            this.dialog = new ProgressDialog(context, 1);
            this.dialog.setMessage("Deleting Donation");
            this.dialog.show();
        }

        @Override
        protected String doInBackground(String... params) {

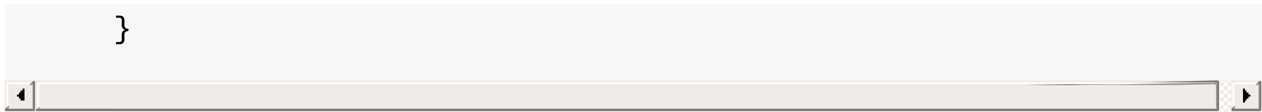
            try {
                return (String) DonationApi.delete((String) para
ms[0], (String) params[1]);
            } catch (Exception e) {
                Log.v("donate", "ERROR : " + e);
                e.printStackTrace();
            }
            return null;
        }

        @Override
        protected void onPostExecute(String result) {
            super.onPostExecute(result);

            String s = result;
            Log.v("donate", "DELETE REQUEST : " + s);

            new GetAllTask(Report.this).execute("/donations");

            if (dialog.isShowing())
                dialog.dismiss();
        }
```



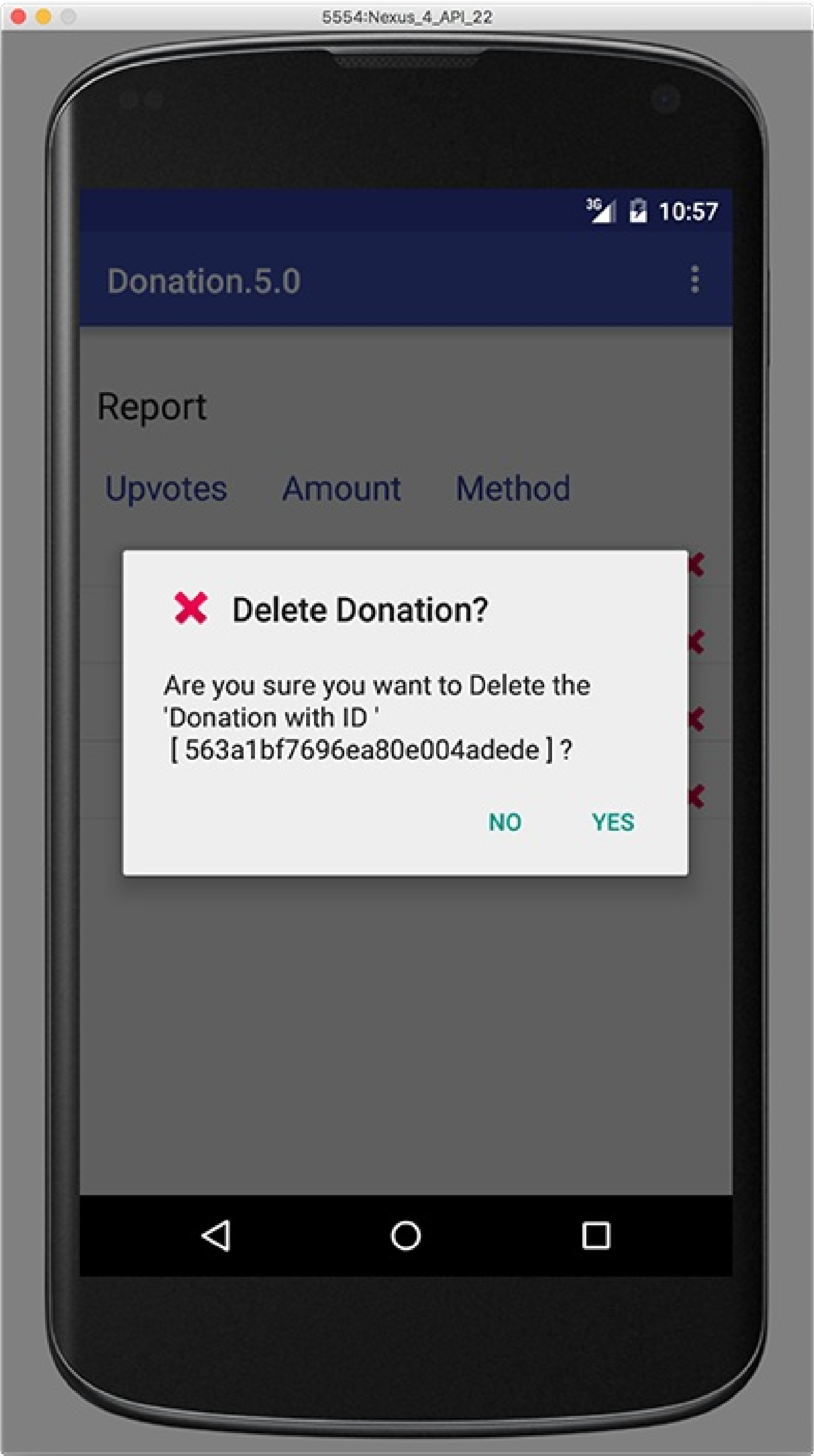
and also this method

```
public void onDonationDelete(final Donation donation) {
        String stringId = donation._id;
        AlertDialog.Builder builder = new AlertDialog.Builder(th
is);
        builder.setTitle("Delete Donation?");
        builder.setIcon(android.R.drawable.ic_delete);
        builder.setMessage("Are you sure you want to Delete the
\'Donation with ID \' \n [ "
                + stringId + " ] ?");
        builder.setCancelable(false);

        builder.setPositiveButton("Yes", new DialogInterface.OnC
lickListener() {
            public void onClick(DialogInterface dialog, int id)
{

                new DeleteTask(Report.this).execute("/donations"
, donation._id);
            }
        }).setNegativeButton("No", new DialogInterface.OnClickLi
stener() {
            public void onClick(DialogInterface dialog, int id)
{
                dialog.cancel();
            }
        });
        AlertDialog alert = builder.create();
        alert.show();
    }
```

Now, see can you call the above **onDonationDelete()** in your **onClick()** method so you get something like this when you click on the delete button


5554:Nexus_4_API_22
10:57
Donation.5.0
Report
Upvotes
Amount
Method
Delete Donation?
Are you sure you want to Delete the 'Donation with ID ' [563a1bf7696ea80e004adede] ?
NO
YES

Donate Activity - Add a Donation

Here's the complete **Report.java** activity before we complete our penultimate step in this lab

```
package ie.app.activities;

import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.os.AsyncTask;
import android.support.v4.widget.SwipeRefreshLayout;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import java.util.List;

import ie.app.R;
import ie.app.api.DonationApi;
import ie.app.models.Donation;

public class Report extends Base implements OnItemClickListener,
OnClickListener {
    ListView listView;
    DonationAdapter adapter;
    SwipeRefreshLayout mSwipeRefreshLayout;
```

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_report);

        listView = (ListView) findViewById(R.id.reportList);
        mSwipeRefreshLayout = (SwipeRefreshLayout) findViewById(
R.id.report_swipe_refresh_layout);

        new GetAllTask(this).execute("/donations");

        mSwipeRefreshLayout.setOnRefreshListener(new SwipeRefres
hLayout.OnRefreshListener() {
            @Override
            public void onRefresh() {
                new GetAllTask(Report.this).execute("/donations"
);
            }
        });

    }

    @Override
    public void onItemClick(AdapterView<?> arg0, View row, int p
os, long id) {

        new GetTask(this).execute("/donations", row.getTag().toS
tring());
    }

    @Override
    public void onClick(View view) {
        if (view.getTag() instanceof Donation) {
            onDonationDelete((Donation) view.getTag());
        }
    }

    public void onDonationDelete(final Donation donation) {
        String stringId = donation._id;
```

```
        AlertDialog.Builder builder = new AlertDialog.Builder(th
is);
        builder.setTitle("Delete Donation?");
        builder.setIcon(android.R.drawable.ic_delete);
        builder.setMessage("Are you sure you want to Delete the
\'Donation with ID \' \n [ "
                + stringId + " ] ?");
        builder.setCancelable(false);

        builder.setPositiveButton("Yes", new DialogInterface.OnC
lickListener() {
            public void onClick(DialogInterface dialog, int id)
{

                new DeleteTask(Report.this).execute("/donations"
, donation._id);
            }
        }).setNegativeButton("No", new DialogInterface.OnClickLi
stener() {
            public void onClick(DialogInterface dialog, int id)
{
                dialog.cancel();
            }
        });
        AlertDialog alert = builder.create();
        alert.show();
    }

    private class GetAllTask extends AsyncTask<String, Void, List
<Donation>> {

        protected ProgressDialog dialog;
        protected Context context;

        public GetAllTask(Context context) {
            this.context = context;
        }

        @Override
        protected void onPreExecute() {
```

```
            super.onPreExecute();
            this.dialog = new ProgressDialog(context, 1);
            this.dialog.setMessage("Retrieving Donations List");
            this.dialog.show();
        }

        @Override
        protected List<Donation> doInBackground(String... params)
 {

            try {
                return (List<Donation>) DonationApi.getAll((Stri
ng) params[0]);
            } catch (Exception e) {
                Log.v("ASYNC", "ERROR : " + e);
                e.printStackTrace();
            }
            return null;
        }

        @Override
        protected void onPostExecute(List<Donation> result) {
            super.onPostExecute(result);

            app.donations = result;
            adapter = new DonationAdapter(context, app.donations
);
            listView.setAdapter(adapter);
            listView.setOnItemClickListener(Report.this);
            mSwipeRefreshLayout.setRefreshing(false);

            if (dialog.isShowing())
                dialog.dismiss();
        }
    }

    private class GetTask extends AsyncTask<String, Void, Donati
on> {

        protected ProgressDialog dialog;
```

```
        protected Context context;

        public GetTask(Context context) {
            this.context = context;
        }

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            this.dialog = new ProgressDialog(context, 1);
            this.dialog.setMessage("Retrieving Donation Details"
);
            this.dialog.show();
        }

        @Override
        protected Donation doInBackground(String... params) {

            try {
                return (Donation) DonationApi.get((String) param
s[0], (String) params[1]);
            } catch (Exception e) {
                Log.v("donate", "ERROR : " + e);
                e.printStackTrace();
            }
            return null;
        }

        @Override
        protected void onPostExecute(Donation result) {
            super.onPostExecute(result);

            Donation donation = result;

            Toast.makeText(Report.this, "Donation Data [ " + don
ation.upvotes + "]\n " +
                    "With ID of [" + donation._id + "]", Toast.L
ENGTH_LONG).show();

            if (dialog.isShowing())
```

```
                dialog.dismiss();
        }
    }

    private class DeleteTask extends AsyncTask<String, Void, Str
ing> {

        protected ProgressDialog dialog;
        protected Context context;

        public DeleteTask(Context context) {
            this.context = context;
        }

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            this.dialog = new ProgressDialog(context, 1);
            this.dialog.setMessage("Deleting Donation");
            this.dialog.show();
        }

        @Override
        protected String doInBackground(String... params) {

            try {
                return (String) DonationApi.delete((String) para
ms[0], (String) params[1]);
            } catch (Exception e) {
                Log.v("donate", "ERROR : " + e);
                e.printStackTrace();
            }
            return null;
        }

        @Override
        protected void onPostExecute(String result) {
            super.onPostExecute(result);

            String s = result;
```

```
            Log.v("donate", "DELETE REQUEST : " + s);

            new GetAllTask(Report.this).execute("/donations");

            if (dialog.isShowing())
                dialog.dismiss();
        }
    }


    class DonationAdapter extends ArrayAdapter<Donation> {
        private Context context;
        public List<Donation> donations;

        public DonationAdapter(Context context, List<Donation> d
onations) {
            super(context, R.layout.row_donate, donations);
            this.context = context;
            this.donations = donations;
        }

        @Override
        public View getView(int position, View convertView, View
Group parent) {
            LayoutInflater inflater = (LayoutInflater) context
                    .getSystemService(Context.LAYOUT_INFLATER_SE
RVICE);

            View view = inflater.inflate(R.layout.row_donate, pa
rent, false);
            Donation donation = donations.get(position);

            ImageView imgDelete = (ImageView) view.findViewById(
R.id.imgDelete);
            imgDelete.setTag(donation);
            imgDelete.setOnClickListener(Report.this);

            TextView amountView = (TextView) view.findViewById(R
.id.row_amount);
            TextView methodView = (TextView) view.findViewById(R
```

```
.id.row_method);
            TextView upvotesView = (TextView) view.findViewById(
R.id.row_upvotes);

            amountView.setText("" + donation.amount);
            methodView.setText(donation.paymenttype);
            upvotesView.setText("" + donation.upvotes);

            view.setTag(donation._id); // setting the 'row' id t
o the id of the donation

            return view;
        }

        @Override
        public int getCount() {
            return donations.size();
        }
    }
}
```

Now, let's look at how we can add a new Donation and 'insert' it into the remote list of Donations, maintained on the Server.

Open your **Donate.java** Activity and have a look at the current implementations of donateButtonPressed() & reset()

```
public void donateButtonPressed (View view)
    {
        String method = paymentMethod.getCheckedRadioButtonId()
== R.id.PayPal ? "PayPal" : "Direct";
        int donatedAmount =  amountPicker.getValue();
        if (donatedAmount == 0)
        {
            String text = amountText.getText().toString();
            if (!text.equals(""))
                donatedAmount = Integer.parseInt(text);
        }
        if (donatedAmount > 0)
        {
            app.newDonation(new Donation(donatedAmount, method, 0
));
            progressBar.setProgress(app.totalDonated);
            String totalDonatedStr = "$" + app.totalDonated;
            amountTotal.setText(totalDonatedStr);
        }
    }

    @Override
    public void reset(MenuItem item)
    {
        app.totalDonated = 0;
        amountTotal.setText("$" + app.totalDonated);
    }
```

They add to our local list of Donations, and reset a simple field - we'll implement the reset feature in the final step so now, the first thing to do, is look at how we can refactor the donateButtonPressed() method to add a donation to our remote list and then update our total.

We're going to need another AsyncTask for this so see if you can complete this 'InsertTask' based on the previous Tasks you've implemented

```
private class InsertTask extends AsyncTask<Object, Void, String>
{
```

```
        protected ProgressDialog dialog;
        protected Context context;

        public InsertTask(Context context)
        {
            this.context = context;
        }

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            this.dialog = new ProgressDialog(context, 1);
            this.dialog.setMessage("Saving Donation....");
            this.dialog.show();
        }

        @Override
        protected String doInBackground(Object... params) {

            String res = null;
            try {
                Log.v("donate", "Donation App Inserting");

            }

            catch(Exception e)
            {
                Log.v("donate","ERROR : " + e);
                e.printStackTrace();
            }
            return res;
        }

        @Override
        protected void onPostExecute(String result) {
            super.onPostExecute(result);


        }
```

```
    }
```

And referring to the lecture material, complete the donateButtonPressed() method that will invoke the above Task.

Donate Activity - Reset/Delete all Donations

The final step in this lab is the ability to delete, or reset, all our Donations.

Remember - once you reset the Donations associated with this version of our Server (donationweb-4-0) Everyones Donations will be deleted, so you'll have to add in more Donations to keep testing your App.

Anyway :)

Let's finish off our Donation 5.0 App by implementing the 'Reset' Menu option.

First thing to do is bring in this AsyncTask

```
private class ResetTask extends AsyncTask<Object, Void, String>
{

        protected ProgressDialog         dialog;
        protected Context                 context;

        public ResetTask(Context context)
        {
            this.context = context;
        }

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            this.dialog = new ProgressDialog(context, 1);
            this.dialog.setMessage("Deleting Donations....");
            this.dialog.show();
        }
```

```
        @Override
        protected String doInBackground(Object... params) {

            String res = null;
            try {
                    res = DonationApi.deleteAll((String)params[0
]);
            }

            catch(Exception e)
            {
                Log.v("donate"," RESET ERROR : " + e);
                e.printStackTrace();
            }
            return res;
        }

        @Override
        protected void onPostExecute(String result) {
            super.onPostExecute(result);

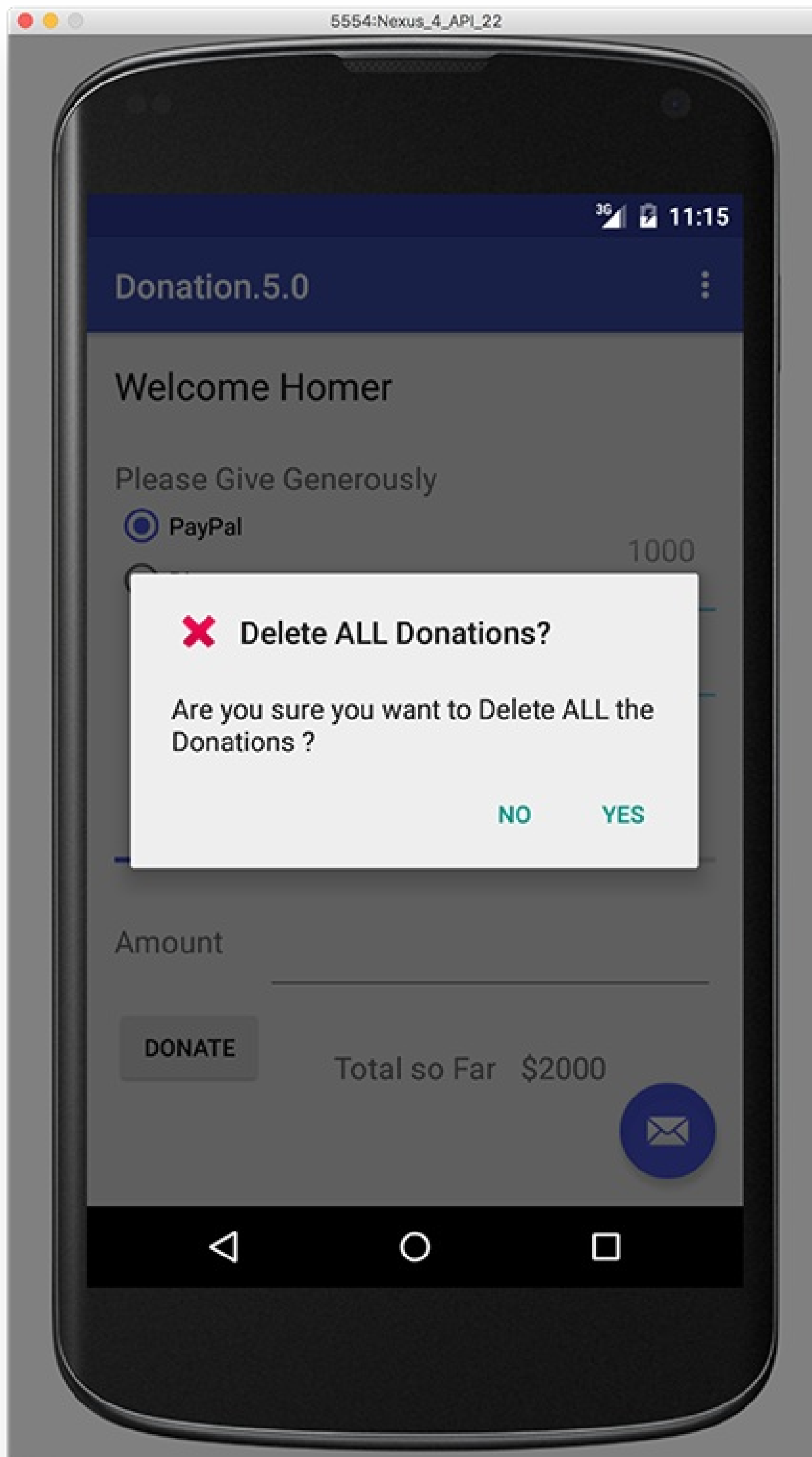
            app.totalDonated = 0;
            progressBar.setProgress(app.totalDonated);
            amountTotal.setText("$" + app.totalDonated);

            if (dialog.isShowing())
                dialog.dismiss();
        }
    }
```

There's really only one line of code which needs to be added to the reset() method so see if you can work out what it is and be able to 'reset' (delete) all the Donations on the Server.

If you get this all working, you'll notice there's a small bug in the app that allows the user to still view the Report Screen, even after we called the reset menu option.

See can you fix this bug and also add in a feature to ask the user if they're sure they want to reset all the donations (similar to the delete a Donation feature on the Report Screen) like so



Solution

This is a solution to the lab:

- Donation.5.0